DISEÑO DE UNA ESTRATEGIA DE VISUALIZACIÓN DE SERVICIOS CON UNA ESTRUCTURA TIPO DASHBOARD EN EL CENTRO DE INVESTIGACIÓN APLICADO Y DESARROLLO EN TECNOLOGÍAS DE LA INFORMACIÓN (CIADTI)

MAYRA ALEJANDRA ROZO BARRERA

UNIVERSIDAD DE PAMPLONA

FACULTAD DE INGENIERIAS Y ARQUITECTURA

DEPARTAMENTO DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA, SISTEMAS, Y TELECOMUNICACIONES

PROGRAMA DE INGENIERÍA DE SISTEMAS

PAMPLONA, 2018

DISEÑO DE UNA ESTRATEGIA DE VISUALIZACIÓN DE SERVICIOS CON UNA ESTRUCTURA TIPO DASHBOARD EN EL CENTRO DE INVESTIGACIÓN APLICADO Y DESARROLLO EN TECNOLOGÍAS DE LA INFORMACIÓN (CIADTI)

MAYRA ALEJANDRA ROZO BARRERA

TRABAJO DE GRADO PRESENTADO PARA OPTAR POR EL TÍTULO DE INGENIERO DE SISTEMAS

Director: MSc. AVILIO VILLAMIZAR

Co-director: Esp. ELVIS NAVARRO VEGA

UNIVERSIDAD DE PAMPLONA

FACULTAD DE INGENIERIAS Y ARQUITECTURA

DEPARTAMENTO DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA, SISTEMAS, Y TELECOMUNICACIONES

PROGRAMA DE INGENIERÍA DE SISTEMAS

PAMPLONA, 2018

Dedicatoria

Este trabajo de grado se lo dedico con todo mi corazón a mi madre, que gracias a su sacrificio y entrega hizo que hoy todo esto sea posible.

A mi padre, que espero que desde el cielo que pueda ver todos mis triunfos y se sienta orgulloso.

Al amor de mi vida, Sandro, que gracias a su apoyo incondicional y a su compañía me motiva día a día a salir adelante.

A mí adorado hermanito, Jesús, para que vea que con esfuerzo y dedicación todo es posible.

Los amo demasiado.

AGRADECIMIENTOS

Gracias a mi madre, que es el motor de mi vida, que ha luchado por sacar adelante a todos sus hijos pese a las adversidades y que me enseño junto a mi padre que en paz descanse la importancia de estudiar y prepararse; gracias a Sandro Antonio Cárdenas, mi novio, porque con su apoyo, compresión y amor he sacado adelante estos cinco años de mi carrera; gracias a mis tíos maternos, que siempre con sus consejos me han ayudado a ir por un buen camino y por estar en cada momento de mi vida. Le agradezco especialmente a mi querido tío Juan, porque siempre ha estado ahí para mí, siendo un ejemplo a seguir.

A los ingenieros del CIADTI en especial al ingeniero Elvis Navarro quien con su amabilidad y experiencia me colaboro en este proceso y al ingeniero Avilio Villamizar por darme la oportunidad de ser parte de esta institución. También quiero agradecer a los docentes del programa de Ingeniera de sistemas, ya que gracias a todos los ellos soy y seguiré siendo una gran profesional. Un especial agradecimiento al profesor Mauricio Rojas, quien me apoyo en la realización de este proyecto, transmitiéndome muchos de sus conocimientos y dedicando parte de su tiempo libre.

¡Muchas gracias!

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

RESUMEN

Diseño de una estrategia de visualización de servicios con una estructura tipo *dashboard* en el Centro de Investigación Aplicado y Desarrollo en Tecnologías de la Información (CIADTI). El CIADTI de la Universidad de Pamplona actualmente trabaja, en las aplicaciones web, con una arquitectura en la cual se hace difícil el mantenimiento y creación de nuevas funcionalidades puesto que el software tiene un alto acoplamiento, además de tener dificultad a la hora de recibir gran cantidad de peticiones en el servidor, dado que la capa de presentación y la capa de la lógica de negocio están embebidas en una sola. En cuanto a la interfaz de usuario en algunas ocasiones se deben realizar muchos pasos para llegar a un objetivo, por ende, esto no facilita la escalabilidad ni la interoperabilidad que son dos características deseables en cualquier sistema. Es por ello, que surge la necesidad de desarrollar una estrategia que permita solucionar estos problemas, ofreciendo la posibilidad de visualizar diversos servicios en un tablero de mando o como es llamado comúnmente en un *dashboard*.

Palabras clave: arquitectura de software, arquitectura orientada a servicios, servicios REst, visualización de servicios, dashboard.

1. INTRODUCCION.

1.1 PLANTEAMIENTO DEL PROBLEMA

Según Merchán[1] los datos son un gran activo en cada una de las organizaciones, pero muchas veces se cree que no son de mayor importancia, sin tomar en cuenta que estos ayudan a la toma de decisiones. En la actualidad se manejan grandes volúmenes de datos, los cuales son un recurso vital y fundamental, ya que con ellos es obtenida la información que puede ser de gran utilidad para cada una de las organizaciones. Se debe tener en cuenta que, del buen manejo de este recurso tan importante, como lo es la información, depende en gran parte el éxito de una organización. La información a lo largo de la historia se ha manejado de diferentes formas, afortunadamente en esta época, se cuenta para este trabajo, con los sistemas de información, los cuales son desarrollados bajo diferentes arquitecturas y diseños, según las necesidades o preferencias de sus usuarios. Los sistemas de información tradicionales muestran cada una de las funcionalidades a las que puede acceder un usuario, sin tener en cuenta la información relevante que le interesa ver o a la cual le gustaría acceder rápidamente, sin la necesidad de dar tantos clics o de buscarla.

Actualmente el Centro de Investigación Aplicado y Desarrollo en Tecnologías de la Información (CIADTI) cuenta con sistemas de información, en los que, si se quiere acceder a cierta información que habitualmente se consulta, el usuario debe ingresar a la funcionalidad específica para este propósito, y buscar en el menú hasta que logre encontrar lo deseado, esta búsqueda, se convierte en una tarea tediosa cuando estos menús tienen demasiadas opciones. Un claro ejemplo que evidencia esto, es el software Academusoft. En el capítulo 5, numeral 5.2, se presenta lo anteriormente dicho, mediante un ejemplo, en el cual se requieren de cinco pasos para lograr la consulta del horario por parte del estudiante.

[1] MERCHÁN, Ana Lucia. Importancia de utilizar Business Intelligence en una organización. Cuenca.: Universidad Tecnológica Israel. 2012. 1 p.

El CIADTI en sus aplicaciones web, trabaja con una arquitectura a tres capas, en la que la primera corresponde a la capa de usuario, la segunda a la capa de presentación y de lógica de negocio y la tercera tiene que ver con la capa de acceso a los datos. Cabe señalar que esta información, fue obtenida mediante la recopilación de la misma, en mesas de trabajo realizadas con los ingenieros del CIADTI. Este tipo de arquitectura implica que el tráfico en la red aumente a medida que sean enviadas una gran cantidad de peticiones a un servidor, se tenga dificultad a la hora de programar nuevas funcionalidades o de modificar las ya existentes, aumente la complejidad del sistema, las aplicaciones no sean fácilmente mantenibles puesto que son altamente acopladas, la realización de pruebas sea compleja y además que se haga imposible la fácil escalabilidad e interoperabilidad. Esto lleva a que sea necesario indagar en temas como las arquitecturas orientadas a servicios, arquitecturas distribuidas, los tableros de mando y tecnologías que brindan la posibilidad de cambiar la estructura de trabajo actual y de solventar los problemas que tiene la arquitectura anteriormente mencionada.

De acuerdo con los argumentos expuestos anteriormente este proyecto pretende dar respuesta a la siguiente pregunta:

¿Qué estrategia se debe utilizar para la visualización de servicios en una estructura tipo *dashboard*?

1.2 JUSTIFICACIÓN

El hecho que se deban dar tantas “vueltas” para encontrar información que constantemente es requerida, por ejemplo, consultas de horario por parte del estudiante durante el semestre, hace que estos procesos repetitivos consuman más recursos de lo que deberían y por ende se vuelvan poco eficientes al ser solicitados por una gran cantidad de usuarios en un tiempo determinado. Esto además de causar lo anteriormente mencionado, ocasiona una grave saturación de los servidores, llevando a que estos dejen de funcionar temporalmente. Es de gran importancia definir una estrategia que permita visualizar servicios, a través de una interfaz, en la cual se defina la información que un usuario desearía ver al ingresar

al sistema. La visualización de servicios debe estar pensada de forma que no importe el lenguaje de programación en los que estén desarrollados, ya sea java, Python, entre otros, debe permitirse una integración de plataformas heterogéneas, en la cual el usuario final no note la diferencia en cuestión de la tecnología en la que fue construido el servicio.

Este proyecto se desarrolló con el fin de diseñar una estrategia de visualización de servicios en estructuras de tipo *dashboard* (en el capítulo 2, numeral 2.2. que corresponde al marco conceptual, se encuentra una definición formal de este término), la cual permita mostrar información relevante para el usuario, y así se conciba ésta como un activo de alto valor en el sistema, además que sea amena la navegabilidad y la experiencia de usuario. Se proyecta que el diseño de esta estrategia sea un punto de partida para la evolución del software del Centro de Investigación Aplicado y Desarrollo en Tecnologías de la Información (CIADTI).

1.3 OBJETIVOS

1.3.1 Objetivo general

Proponer el diseño de una estrategia de visualización de servicios con una estructura tipo *dashboard*, para el sistema de información de la Universidad de Pamplona

1.3.2 Objetivos específicos

1. Elaborar un marco de referencia de arquitecturas orientadas a servicios (SOA) y visualización de servicios en una interfaz de tipo *dashboard*.
2. Definir la arquitectura lógica para la visualización de servicios en una interfaz de tipo *dashboard*
3. Validar la estrategia de visualización de servicios en un prototipo tipo *dashboard* con el fin de dar soporte a una funcionalidad base

1.4 METODOLOGIA

La investigación usa un paradigma cualitativo en donde se busca el desarrollo de una nueva estrategia de visualización, de la cual no se cuenta con mucha información para su construcción, por lo que se usan varias sesiones en mesas de trabajo con el equipo de desarrollo para determinar la arquitectura y estructura de trabajo. Esta tiene un alcance exploratorio ya que el objetivo, es la investigación y desarrollo de un tema sobre el cual no se tienen conocimientos. Después de esta investigación el prototipo resultante pasara por un proceso, en el que el CIADTI evaluara si es factible su utilización e implementación, para uso de toda la comunidad universitaria, desde administrativos hasta los estudiantes.

1.5 FUENTES DE INFORMACIÓN

Se investigó en fuentes virtuales de información, como monografías, bases de datos bibliográficas, trabajos de grado disponibles en la web, base de datos de la Universidad de Pamplona, además se contó con el apoyo y experiencia de los ingenieros dentro del CIADTI.

1.6 CONSIDERACIONES ÉTICAS

Parte de los datos usados en el proceso de desarrollo de este proyecto fueron proporcionados por el CIADTI, información que pertenece a sus bases de datos, la cual fue manejada con el mayor cuidado y discreción, ya que, esta es de su propiedad y no puede ser usada para otros fines ajenos a este proyecto.

2 MARCO DE REFERENCIA

2.1 ANTECEDENTES

2.1.1 Ámbito internacional

- **Título:** *Dashboard* para la gestión y monitorización de cuentas del servicio *cloud* AWS.

  Autor: Carlos Santamaría Barroso

  Lugar: Escuela Técnica Superior de Ingeniería (ICAI), Madrid, España

  Fecha: julio 2017

  Link: http://hdl.handle.net/11531/26339

  Palabras clave: AWS, *dashboard*, *cloud*, monitorización, pago por uso.

  Resumen:

  “En este proyecto se ha desarrollado una aplicación web destinada a la monitorización de todos los posibles servicios cloud que tenga la entidad colaboradora, centrándonos solo en uno (AWS), con el fin de facilitar ciertas tereas de gestión, así como a ayudar a tener una idea más concreta y precisa de todos los datos de importancia dentro de estos servicios.”[2]

- **Título:** Metodología para el diseño de Dashboards orientado hacia el registro de evidencias en el proceso de evaluaciones institucionales.
  Autor: Daniel Andrés Martínez Robalino
  Lugar: Universidad Internacional de la Rioja, Puyo, Ecuador.
  Fecha: 24 de julio de 2017
  Link: https://reunir.unir.net/bitstream/handle/123456789/6171/MARTINEZ%20ROBALINO%2C%20DANIEL%20ANDRES.pdfsequence=1&isAllowed=y

[2] SANTAMARÍA, Carlos. Dashboard para la gestión y monitorización de cuentas del servicio cloud AWS. Grado en ingeniería telemática. Madrid.: Escuela técnica superior de ingeniería. 2017. 10 p.

Palabras clave: evaluación institucional, evidencias, metodología, registros, tablero de control.

Resumen:

En la actualidad, la evaluación institucional desempeña un papel fundamental en las instituciones educativas que desean alcanzar estándares de calidad, a través del cumplimiento de objetivos y metas en un periodo determinado.
La evaluación institucional permite a las instituciones educativas hacer un diagnóstico sobre el estado actual de la calidad académica propuesta en su proyecto educativo y en plan de estudios.
En este sentido, la aplicación de metodologías a través de herramientas que permitan desarrollar software para el seguimiento y evaluación de indicadores basados en evidencias, constituyen un papel fundamental al momento de conseguir una certificación o acreditación que asegure su aval por los máximos órganos rectores.
A través del presente trabajo se proporciona una metodología, que permita a los desarrolladores de software diseñar tablero de control (dashboards), basados en elementos comunes, y que cumplan con diseños de usabilidad, orientado hacia el registro de evidencias en procesos de evaluación institucional. [3]

2.1.2 Ámbito nacional.

- **Título:** Estado del Arte: Servicios Web

Autores: Carlos Andrés Morales Machuca.

Lugar: Universidad Nacional de Colombia. Bogotá, Colombia

Link: http://www.bivica.org/upload/doc1.pdf

Palabras clave: web services, servicios web, UDDI, WSDL,

SOAP, SOA.

Resumen:

Los servicios web son sistemas de software que permiten el intercambio de datos y funcionalidad entre aplicaciones sobre una red. Esta soportado en diferentes estándares que garantizan la interoperabilidad de los servicios. Los

[3] MARTÍNEZ, Daniel. Metodología para el diseño de Dashboards orientado hacia el registro de evidencias en el proceso de evaluaciones institucionales. Trabajo de fin de Master. Puyo.: Universidad Internacional de la Rioja. 2017. 2 p.

servicios web utilizan como su gran insumo el lenguaje extensible de marcado XML y se basa en una arquitectura en la que se define el servicio web a través de uno de los lenguajes estándar se publica en un directorio donde se halla la descripción anteriormente hecha y se utiliza de acuerdo a las expectativas de resolver una necesidad de acuerdo con la descripción provista. La arquitectura que mejor se ha adaptado al mundo de los servicios web es SOA brindando un enfoque que ha adoptado los negocios y ha incrementado el intercambio electrónico de datos y el comercio electrónico. Se plantea como problema la ausencia de arquitecturas que permitan a los empresarios medianos y pequeños ingresar sus organizaciones al esquema de Orientación a Servicios y Procesos de Negocios.[4]

2.1.3 Ámbito local

- **Título:** Una arquitectura de software para la integración de objetos de aprendizaje basada en servicios web.

  Autores: Mauricio Rojas C. y Jonás Montilva

  Lugar: 9th Latin American and Caribbean Conference for Engineering and Technology. Medellín, Colombia.

  Fecha: agosto 3-5 de 2011

  Link: http://laccei.org/LACCEI2011-Medellin/RefereedPapers/ELDE120_Rojas.pdf

  Palabras clave: arquitectura de software, objetos de aprendizaje, SCORM, LMS, repositorios de objetos de aprendizaje, servicios web.

  Resumen:

  Los avances en los entornos de Educación Virtual han permitido optimizar los procesos de elaboración, distribución y acceso de contenidos a través de la reutilización de objetos de aprendizaje, los cuales se encuentran alojados en repositorios. Con el fin de garantizar la interoperabilidad, integración y reusabilidad de los objetos de aprendizaje se han diseñado estándares, como SCORM, que garantizan la interoperabilidad estructural de los objetos de aprendizaje. Adicionalmente, existen los Sistemas de Gestión de Aprendizaje Virtual (Learning Management Systems - LMS) que permiten diseñar

[4] MORALES, Carlos. Estado del arte: Servicios Web. Bogotá.: Universidad Nacional de Colombia. 1 p.

contenidos para plataformas E-Learning, los cuales admiten reutilizar objetos de aprendizaje que previamente han sido seleccionados en otros sistemas de tipo repositorio. Los repositorios de aprendizaje y los LMS, en la mayoría de los casos, funcionan como dos sistemas independientes. Como una alternativa de integración, en este artículo se describe una arquitectura de software basada en servicios web orientada a la construcción de sistemas de gestión de E-Learning integrales con dos funcionalidades principales: Integración de búsquedas de objetos de aprendizaje en repositorios distribuidos y construcción de contenidos para plataformas E-Learning basado en reutilización de objetos de aprendizaje**.** [5]

- **Título:** Análisis comparativo entre la arquitectura modelo vista controlador (MVC) y la arquitectura orientada a servicios (SOA) en el desarrollo del sistema de información del fondo de empleados de la universidad de pamplona (FEUP).

  Autor: Wilson Arley Gomez Jaimes

  Lugar: Universidad de Pamplona, Pamplona, Colombia.

  Fecha: 2012

  Link: http://serviciosacademicos.unipamplona.edu.co/prestamo/

  Resumen:

  En el entorno actual de cambios constantes, las organizaciones están obligadas a emprender una búsqueda continua de nuevos métodos para consolidar ventajas competitivas sostenibles.

  La presente investigación tiene como objetivo realizar un estudio comparativo entre las arquitecturas de software SOA (Arquitectura Orientada a Servicios) y MVC (Arquitectura Modelo Vista Controlador), orientado al desarrollo del Sistema de información del Fondo De Empleados De La Universidad De Pamplona.

  Para comenzar con el estudio comparativo de las arquitecturas, se utilizó como punto de partida una combinación de métodos de evaluación de arquitecturas de software como: Bosch (2000), Losavio (2003) y Architecture Trade-off Analysis Method (ATAM), basados en este estudio la arquitectura que mejor se ajusta a los atributos de calidad que debe poseer el sistema es la de MVC que se aplicará como base para el diseño arquitectónico del

[5] ROJAS, Mauricio y MONTILVA, Jonás. Una arquitectura de software para la integración de objetos de aprendizaje basada en servicios web. Medellin.: 9th Latin American and Caribbean Conference for Engineering and Technology, 2011. 1 p.

Sistema Información para el Fondo De Empleados De La Universidad De Pamplona, el mismo que al realizar un análisis de beneficios mejora la eficiencia de las Secretarias y sus Asociados de tal manera que pueden llevar a cabo la mayor parte de los procesos en línea los cuales se hacían anteriormente de forma personal mejorando los tiempos de respuesta, buscando así el aseguramiento de la calidad en el desarrollo del proyecto, la disminución o mitigación de los riesgos y la facilidad para definir los entregables de dicho proyecto.

Finalmente, en la propuesta del proyecto se detalla un enfoque al despliegue de la aplicación en su entorno real [6]

2.2 MARCO CONCEPTUAL

Acoplamiento: "es la fuerza de las dependencias entre dos subsistemas",[7] otra definición acertada es la siguiente: **"**es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas"[8]. Bruegge y Dutoit[9] indican en su libro Ingeniería de software orientado a objetos que una característica deseable en todo sistema es que sus componentes se encuentren débilmente acoplados, puesto que, al estar de esta manera son independientes unos de otros, impidiendo así que, las modificaciones de uno afecten a los demás.

Aplicación web: "una aplicación web es una aplicación o herramienta informática accesible desde cualquier navegador, bien sea a través de internet (lo habitual) o bien a través de una red local,"[10] es decir, que es un software desarrollado para ser ejecutado en los navegadores web.

---

[6] GOMEZ, Wilson. Análisis comparativo entre la arquitectura modelo vista controlador (MVC) y la arquitectura orientada a servicios (SOA) en el desarrollo del sistema de información del fondo de empleados de la universidad de pamplona (FEUP). Trabajo de grado para optar por el título de ingeniero de sistemas. Pamplona.: Universidad de Pamplona. 2012. 6 p.

[7] BRUEGGE, Bernd y DUTOIT, Allen. Ingeniería de software orientado a objetos. Pearson educación.: México, 2002. 198 p.

[8] VISCONTI, Marcello y ASTUDILLO, Hernán. Fundamentos de ingeniería de software [en línea]. Universidad Técnica Federico Santa María. [consultado: 12 de octubre de 2018]. Disponible en internet: http://roa.ult.edu.cu/bitstream/123456789/401/1/08-Patrones.pdf

[9] BRUEGGE, Bernd y DUTOIT, Allen. Ingeniería de software orientado a objetos. Pearson educación.: México, 2002. 198 p.

[10] NEOSOFT. ¿Qué es una aplicación web? [en línea]. Neosoft. [8 de enerio de 2018]. [Consultado: 9 de noviembre de 2018]. Disponible en internet: https://www.neosoft.es/blog/que-es-una-aplicacion-web/

Arquitectura de software: según la IEEE una arquitectura de software es "la organización fundamental de un sistema, compuesta por sus componentes, las relaciones entre ellos y su ambiente y los principios que gobiernan su diseño y evolución."[11] En otras palabras, se encarga de especificar la estructura y organización que tendrá el sistema.

Backend: conocido como el lado del servidor. "Es la parte de la app que el usuario final no puede ver. Su función es acceder a la información que se solicita, a través de la app, para luego combinarla y devolverla al usuario final,"[12] es decir, este se encarga de toda la lógica que permite que funcionen las aplicaciones. Para la construcción de esta logica se utilizan diferentes lenguajes de programación como Java, Php, Python, y demás.

Base de datos: "una base de datos no es más que un conjunto de información (un conjunto de datos) relacionada que se encuentra agrupada o estructurada."[13] Este almacén de datos se encarga de guardar toda la información concerniente a una aplicación.

Casos de uso: "un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema."[14] En otras palabras, los casos de uso

---

[11] ISO/IEC Std. 42010: Systems and software engineering – Architectural description. (2009). Citado por: JAÉN, Juan, ROMERO, José y VALLECILLO, Antonio. Especificación de descripciones arquitectónicas multivista basada en modelos [en línea]. Vol. 3, No. 2, 2009. 2 p. Disponible en internet: http://www.lcc.uma.es/~av/Publicaciones/09/DSDM09-MM.pdf

[12] MURILLO, María. Todo lo que necesitás saber sobre backend | All you need to know regarding Backend [en línea]. MMA Mobile Marketing Association. [6 de enero de 2016]. [Consultado: 9 de noviembre de 2018]. Disponible en internet: https://www.mmaglobal.com/news/todo-lo-que-necesitas-saber-sobre-backend-all-you-need-know-regarding-backend

[13] GOMEZ, María del Carmen. Notas del curso bases de datos [en línea]. Universidad Autónoma Metropolitana. México. 2013. 8 p. [Consultado: 9 de noviembre de 2018]. Disponible en internet: http://cua.uam.mx/pdfs/conoce/libroselec/Notas_del_curso_Bases_de_Datos.pdf

[14] ECURED. Caso de uso [en línea]. ECURED. [12 de febrero de 2016]. [Consultado: 9 de noviembre de 2018]. Disponible en internet: https://www.ecured.cu/Caso_de_uso

permiten especificar la comunicación entre los actores y el sistema. Generalmente se hace uso de ellos para capturar los requerimientos funcionales del sistema.

Cliente: un cliente en el ámbito de sistemas se puede definir como "una aplicación informática o un ordenador que consume un servicio remoto en otro ordenador conocido como servidor, normalmente a través de una red de telecomunicaciones."[15] Este envía peticiones al servidor para obtener la información deseada.

***Dashboard*:** un tablero de mando o en ingles d*ashboard.* Es "un conjunto de indicadores esenciales organizados de manera tal que pueden reconocerse fácilmente patrones de rendimiento"[16]. Su función principal es visualizar los principales indicadores de una empresa mediante una interfaz llamativa y graficas que permitan convertir los datos en información.

***Framework*:** También llamado marco de trabajo. Este término hace referencia a "una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación"[17], es decir, que cuenta con un conjunto de técnicas, conceptos y criterios, que permiten la construcción de software.

Front-end: conocido como el lado del cliente. "Es la parte del desarrollo web que se dedica de la parte frontal de un sitio web, en pocas palabras del diseño de un sitio web, desde la estructura del sitio hasta los estilos como colores, fondos, tamaños hasta llegar a las animaciones y efectos"[18], este se encarga de todo lo

---

[15] Sadoski, Darleen. Client/Server Software Architectures--An Overview, Software Technology Roadmap, 1997-08-02. Retrieved on 2008-09-16. Citado por: WIKIPEDIA. Cliente (informática) [en línea]. WIKIPEDIA La enciclopedia libre. [Consultado: 9 de noviembre de 2018]. Disponible en internet: https://es.wikipedia.org/wiki/Cliente_(inform%C3%A1tica)#cite_note-1

[16] PUENTE, Raquel. Un tablero de mando para mercadeo [en línea]. En: Debates IESA. 2008, vol. XIII, no. 4, p 74-80. [Consultado: 10 de mayo de 2018]. Disponible en internet: http://servicios.iesa.edu.ve/portal/Articulos/14-Puente-Untablerodemando.pdf

[17] GUTIÉRREZ, Javier. ¿Qué es un framework web? [en línea].1 p. [Consultado: 9 de noviembre de 2018]. Disponible en internet: http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf

[18] FALCONMASTERS. Explicando que es Front-End, que es Back-End y sus características [en línea]. FM FalconMasters. [10 de septiembre de 2014]. [Consultado: 9 de noviembre de 2018].

relacionado con la interfaz de usuario. Generalmente se construye mediante tres tecnologías, HTML (*HyperText Markup Language*), CSS (*Cascading Style Sheets*) y JavaScript.

Interfaz de usuario: es la parte visible de un software, la cual permite la interacción del usuario con el sistema. Una definición más formal de esta es: la interfaz de usuario es "el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora y comprende todos los puntos de contacto entre el usuario y el equipo."[19]

Requerimiento de software: los requerimientos o requisitos de software "describen los servicios que ha de ofrecer el sistema y las restricciones asociadas a su funcionamiento."[20]

RFC: estas siglas en ingles corresponden a *Request For Comments*, que en español quiere decir solicitud o petición de comentarios. Según Pillou[21] son documentos que se encargan de definir, describir y estandarizar tecnologías nuevas o ya existentes, relacionadas con internet y con redes en general. Estas son una referencia para la comunidad de internet. Por ejemplo, el RFC 2616 especifica el protocolo HTTP y el RFC 8141 define el estándar URN.

Servicio web: IBM define los servicios web como "aplicaciones autónomas modulares que se pueden describir, publicar, localizar e invocar a través de una red."[22] Existen diferentes estilos de servicios web entre ellos se tiene RPC (*Remote*

Disponible en internet: http://www.falconmasters.com/web-design/que-es-front-end-y-que-es-back-end/

[19] LEON, Fran. ¿Qué es una interfaz de usuario? ¿Qué elementos debe contener? [en línea]. Merca2.0. [27 de septiembre de 2015]. [Consultado: 9 de noviembre de 2018]. Disponible en internet: https://www.merca20.com/que-es-una-interfaz-de-usuario-que-elementos-debe-contener/

[20] DECSAI. Especificación de requerimientos: Diseño de bases de datos [en línea]. Departamento de ciencias de la computación e I.A. Universidad de Granada. 4 p. [Consultado: 10 de noviembre de 2018]. Disponible en internet: https://elvex.ugr.es/idbis/db/docs/design/2-requirements.pdf

[21] Equipo CCM. RFC (petición de comentarios) [en línea]. CCM. [Consultado 10 de noviembre de 2018]. Disponible en internet: https://es.ccm.net/contents/276-rfc-peticion-de-comentarios

[22] IBM. Servicios web. . IBM Knowledge Center [en línea]. [Consultado: 20 de septiembre de 2018]. Disponible en internet:

Procedure Call), REST (*Representational State Transfer*) y SOAP (*Simple Object Access Protocol*).

Servidor: "el término servidor tiene dos significados en el ámbito informático. El primero hace referencia al ordenador que pone recursos a disposición a través de una red, y el segundo se refiere al programa que funciona en dicho ordenador."[23] Otra definición de servidor es: "un servidor es un ordenador u otro tipo de equipo informático encargado de suministrar información a una serie de clientes."[24]

Sistema legado: también conocidos como sistemas heredados. "Un sistema legado es un componente técnicamente obsoleto de un entorno de gestión de contenido,"[25] es decir, que son aquellos sistemas que se mantienen dentro de una organización y que son difíciles de cambiar o actualizar.

Software: el termino software se puede definir como un "conjunto de datos y programas que maneja el ordenador. Es la parte lógica o inmaterial de un sistema informático."[26] Este permite el cumplimiento de tareas específicas en el sistema.

W3C: *World Wide Web* por sus siglas en ingles. Es una asociación de organizaciones que se encarga de proveer estándares web. Esta se define a sí misma como: "el Consorcio World Wide Web (W3C) es un consorcio internacional

---

https://www.ibm.com/support/knowledgecenter/es/SS7K4U_9.0.0/com.ibm.websphere.zseries.doc/ae/cwbs_wbs2.html

[23] Digital Guide. ¿Qué es un servidor? [en línea]. Digital Guide. [30 de mayo de 2016]. [Consultado: 10 de noviembre de 2018]. Disponible en internet: https://www.ionos.es/digitalguide/servidores/know-how/que-es-un-servidor-un-concepto-dos-definiciones/

[24] LATRE, Esteban. Qué es un servidor y para qué sirve [en línea]. 2016. [Consultado: 20 de septiembre de 2018]. Disponible en internet: https://infortelecom.es/blog/que-es-un-servidor-y-para-que-sirve/

[25] CONDE, Juan. Sistemas legados. Benemérita Universidad Autónoma de Puebla. [Consultado: 10 de noviembre de 2018]. Disponible en internet: http://climate.cs.buap.mx/condejc/cursos/Material/WebServices_/Notas/06%20Sistemas%20Legados.pdf

[26] RODRIGUEZ-ARAGON, Liceo. Tema 3: Software: Sistemas operativos y aplicaciones [en línea]. Universidad Rey Juan Carlos. [Consultado: 10 de noviembre de 2018]. Disponible en internet: https://previa.uclm.es/profesorado/licesio/docencia/ib/ibtema3a.pdf

donde las organizaciones miembro, personal a tiempo completo y el público en general, trabajan conjuntamente para desarrollar estándares Web."[27]

***Widgets*:** son los componentes que conforman el *dashboard*. Un *widget* hace referencia a "pequeñas aplicaciones que tienen como principal cometido mostrar y dar fácil acceso a algunas de las principales funciones de un sistema o terminal"[28]

***Wireframe*:** se puede definir como un "boceto donde se representa visualmente, de una forma muy sencilla y esquemática la estructura de una página web."[29] Esta representación se realiza en escala de grises y por lo general se utiliza en los inicios del proyecto como maqueta de una aplicación.

2.3 MARCO TEÓRICO

2.3.1 Arquitectura orientada a servicios (SOA)

Este concepto no es nuevo, ya que, en la década de los noventa fue descrito por primera vez por Gartner (*), pero en la actualidad ha tomado fuerza gracias a los servicios web. Para empezar a hablar acerca de una arquitectura orientada a servicios se deben tener en cuenta ciertos términos:

Servicio: Según Díaz "Un servicio es un pedazo de funcionalidad que ejecuta de manera aislada e independiente una funcionalidad específica dentro del proceso de negocio. Es invocado a través de una interfaz y su diseño debe cumplir con el principio de reusabilidad."[30], es decir, que un servicio es reutilizable, es independiente de la tecnología en la cual es construido y no depende de la ejecución

---

[27] W3C. Sobre el World Wide Web Consortium [en linea]. W3C World Wide Web. [Consultado: 10 de noviembre de 2018]. Disponible en internet: https://www.w3c.es/Consorcio/about-w3c.html

[28] https://andro4all.com/2013/12/widget

[29] WEBDESDECERO. Wireframes: que son y cómo crearlos [en línea]. [Consultado: 20 de septiembre de 2018]. Disponible en internet: https://webdesdecero.com/wireframes-que-son-y-como-crearlos/

[30] DIAZ, Marinela. Arquitectura orientada a servicios. La Habana.: 14 convención científica de ingeniería y arquitectura, 2008. 2 p.

(*) Es una empresa de consultoría e investigación de tecnologías de la información la cual en 1996 describió SOA en "Service Oriented Architecture, part 1" y "Service Oriented Architecture, part 2".

de otras funciones. Además, se debe tener en cuenta que un servicio recibe una llamada, es decir que es consumido por un cliente y al igual retorna una respuesta

Orquestación: Este concepto se basa en la lógica y coordinación utilizada para hacer uso de los servicios.

Consumidor: Es el cliente el cual puede ser una función o un proceso, el cual consume un servicio.

Proveedor: Como su nombre lo indica es aquel que provee un servicio, este puede ser una función o un proceso

Sin estado: Un servicio no depende de la ejecución de otras funciones. Al ser consumido recibe la información necesaria para procesar la respuesta

Con estos conceptos claros, se puede empezar a definir lo que es una arquitectura orientada a servicios. SOA, por sus siglas del inglés, *Service Oriented Architecture*, la cual se caracteriza por unir los objetivos de las organizaciones con el sistema de información. Esta presenta una estructura que permite la realización de aplicaciones heterogéneas, esto permite que se integren aplicaciones desarrolladas en diferentes tecnologías.

Según Díaz “la arquitectura orientada a servicios se refiere a la capacidad de implementar una arquitectura que esté soportada por la creación de servicios que satisfagan los requerimientos del usuario”[31], esto quiere decir que esta arquitectura es una representación de las actividades reales de una organización por medio de servicios, los cuales deben cumplir con un objetivo o resultado específico.

IBM[32] dice que una arquitectura orientada a servicios se fundamenta en 4 principios, los cuales son:

[31] Ibid.
[32] IBM. Service-oriented architecture. IBM Knowledge Center [en línea], 11 de abril de 2018. [consultado: 15 de mayo de 2018]. Disponible en internet: https://www.ibm.com/support/knowledgecenter/es/SSFTDH_7.5.1/com.ibm.wbpm.wid.main.doc/prodoverview/topics/csoa.html

- Acoplamiento débil entre componentes. (Este término es definido en el numeral anterior: 2.2 Marco conceptual).
- Configuración de los componentes, lo que implica que estos componentes se pueden agregar, eliminar o modificar en cualquier momento.
- Integración de componentes, estén desarrollados o no, en diferentes tecnologías.
- Independencia en la ubicación de los componentes, lo que permite crear aplicaciones distribuidas.

Estos principios marcan una gran diferencia con otras arquitecturas y provee de gran cantidad de ventajas tanto corporativas como desde el punto de vista de los departamentos de tecnologías de información.

2.3.1.1 Características

Algunas características de una arquitectura orientada a servicios (SOA) son:

- Los servicios son construidos de tal forma que estos puedan ser reutilizados. Esto permite el aumento de la integración con sistemas legados.
- SOA está basada en estándares, lo que permite que el uso de los servicios sea independiente de la plataforma.
- Es preferible que los servicios construidos sean de alta calidad
- Los servicios deben ser relevantes, esto indica que deben dar como resultado algo significativo.
- Los servicios encapsulan la lógica en la que son construidos, además de tener control sobre esta. Esto implica que los servicios sean autónomos.
- Entre los diferentes servicios existe un acoplamiento bajo puesto que estos no dependen de los otros para su funcionamiento.
- Para lograr funcionalidades más profundas se puede realizar una orquestación de servicios.
- Los servicios reducen el consumo de recursos por que estos no guardan información de sesión lo que lleva a que sean servicios sin estado.

- A la hora de consumir los servicios no debe importar la ubicación de estos, esto permite la construcción de sistemas distribuidos.

2.3.1.2 Ventajas

Son muchas las ventajas que trae una arquitectura orientada a servicios, entre ellas se pueden destacar:

- Integración de aplicaciones construidas en diferentes tecnologías
- Facilidad a la hora de realizar cambios en los servicios, ya sea agregar, eliminar o modificar estos. Esto gracias al acoplamiento débil
- Interoperabilidad
- Reutilización de servicios para crear nuevas aplicaciones, es decir, un servicio sirve para n aplicaciones
- Aplicaciones más productivas y manejables
- Sistemas altamente escalables

Además, poner en práctica esta arquitectura, resulta en ventajas competitivas a las organizaciones, algunas de ellas son:

- La forma en que trabaja incrementa la eficiencia.
- Su mantenimiento es más sencillo, por lo tanto, reduce costos.
- Ayuda a la toma de decisiones, ya que al integrar las aplicaciones de una organización hace que la información sea exacta y actualizada.

2.3.2 Servicios web

Primero algo de historia de los servicios web

> Los servicios Web fueron "inventados" para solucionar el problema de la interoperabilidad entre las aplicaciones. Al principio de los 90, con el desarrollo de Internet/LAN/WAN, apareció el gran problema de integrar aplicaciones diferentes. Una aplicación podía haber sido desarrollada en C++ o Java, y ejecutarse bajo Unix, un PC, o un computador mainframe. No había una forma fácil de intercomunicar dichas aplicaciones. Fue el desarrollo de XML el que hizo posible compartir datos entre aplicaciones con diferentes plataformas hardware a través de la red, o incluso a través de Internet. La razón de que se llamasen servicios Web es que fueron diseñados para residir en un servidor Web, y ser llamados a través de Internet, típicamente vía protocolos HTTP, o HTTPS. De esta forma se asegura que un

> servicio puede ser llamado por cualquier aplicación, usando cualquier lenguaje de programación, y bajo cualquier sistema operativo, siempre y cuando, por supuesto, la conexión a Internet esté activa, y tenga un puerto abierto HTTP/HTTPS, lo cual es cierto para casi cualquier computador que disponga de acceso a Internet.[33]

Un servicio web o en ingles web service (WS) se puede definir como la comunicación entre aplicaciones. Esta comunicación permite el intercambio de datos entre estas sin tener en cuenta el lenguaje de programación o las tecnologías en que son construidas. World Wide Web Consortium[34] (W3C) define un servicio web como un sistema de software planteado para permitir la interacción interoperable de maquina a máquina a través de una red. Otra definición similar a esta es "Un web service es una vía de intercomunicación e interoperabilidad entre máquinas conectadas en Red"[35]

Estas definiciones tienen que ver con diferentes tipos de sistemas, pero el más usado es frecuentemente el cliente que se comunica con el servidor, siguiendo el estándar SOAP.

2.3.2.1 Características

Algunas características de los servicios web son:

- Interoperabilidad, las aplicaciones que usan estos servicios están construidas en diferentes tecnologías. Esta interoperabilidad se logra a través del uso de estándares abiertos.
- Se pueden usar de forma individual o en conjunto. Esto dependerá de la función que se pretenda llevar a cabo.

---

[33] DEPARTAMENTO DE CIENCIA DE LA COMPUTACION E INTELIGENCIA ARTIFICIAL. Introducción a los Servicios Web. Invocación de servicios web SOAP. Universidad de Alicante [en línea]. 26 de junio de 2014 [consultado: 30 de julio de 2018]. Disponible en internet: http://www.jtech.ua.es/j2ee/publico/servc-web-2012-13/sesion01-apuntes.html

[34] W3C WORKING GROUP NOTE. Web Services Architecture. W3C [en línea]. 11 de febrero de 2004 [consultado: 24 de julio de 2018]. Disponible en internet: https://www.w3.org/TR/ws-arch/#whatis

[35] BAQUERO, José. ¿Qué son los web services y qué tecnología usar en su desarrollo? ARSYS [en línea]. 3 de agosto de 2015 [consultado: 24 de julio de 2018]. Disponible en internet: https://www.arsys.es/blog/programacion/diseno-web/web-services-desarrollo/

- Son débilmente acoplados entre si lo que permite la fácil mantenibilidad. Si se modifica un servicio web no afecta a los demás.
- Permiten que las aplicaciones sean distribuidas, esto implica que no importa el lugar geográfico en el que se encuentren. Esta es la razón de que sean la base de la computación distribuida
- Usan protocolos y estándares basados en texto que facilitan el acceso a su contenido
- Accesibilidad, los servicios web deben ser accesibles a través de internet, para esto hacen uso del protocolo http
- Otra característica importante es que "aportan gran independencia entre la aplicación que usa el servicio Web y el propio servicio"[36]

2.3.2.2 Ventajas

- Fácil construcción e implementación.
- Reducción significativa en el consumo de recursos
- Permite que los sistemas sean escalables
- Fácil mantenibilidad
- Permite la integración de sistemas heterogéneos

2.3.2.3. Estilos de uso

Actualmente se pueden encontrar diferentes estilos de servicios web, pero los más comunes son:

2.3.2.2.1 Remote Procedure Calls (RPC)

Este estilo hoy en día está muy extendido, puesto que cuando surgieron los servicios web, las herramientas se basaban en este, en las llamadas a procedimientos remotos o en inglés Remote Procedure Calls (RPC), además algunos piensan que esta visión es la primera generación de los servicios web. Una característica importante al momento de decidir si adoptar esta técnica o no, es que

[36] WIKIPEDIA. Servicio web. WIKIPEDIA La enciclopedia libre [en línea]. [consultado: 28 de julio de 2018]. Disponible en internet: https://es.wikipedia.org/wiki/Servicio_web

presenta un alto acoplamiento y depende del lenguaje de programación que se use, razón por la cual diversos expertos en el tema piensan que este estilo debe desaparecer.

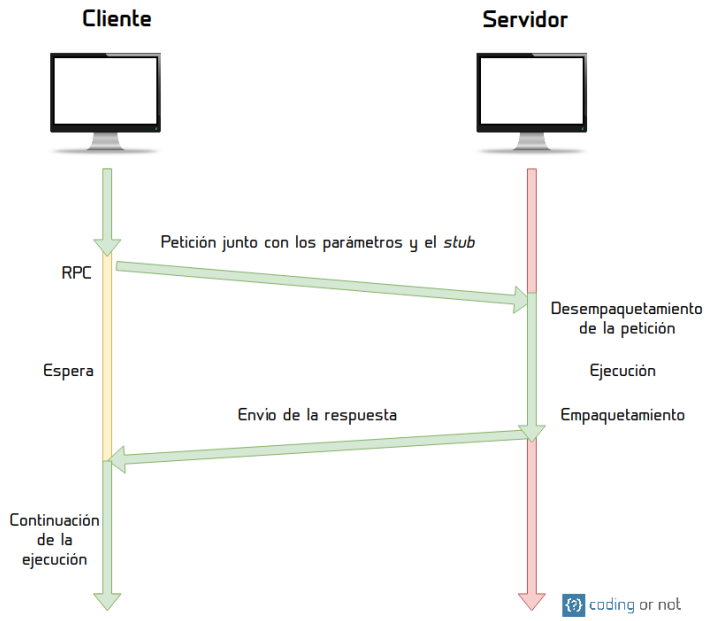
A continuación, se presenta una breve explicación del funcionamiento de RPC o en español, llamada a procedimiento remoto:

1. El cliente hace la llamada al procedimiento remoto mediante un mensaje a través de la red. Este se detiene ya que es un proceso síncrono, es decir, necesita una respuesta del servidor para poder continuar su ejecución. En esta llamada se incluye un stub (o resguardo en español, el *stub* es la pieza de código que le permite al servidor ejecutar la tarea que se le asignó) el cual se encarga de ajustar parámetros y direcciones de memoria de un ambiente (el cliente) a otro (el servidor).
2. El servidor recibe la petición y desempaqueta el mensaje para extraer la información necesaria para realizar la tarea. El stub ayuda a que el servidor sea capaz de convertir parámetros de una representación a otra (de ser necesario), para traducir direcciones de memoria de cliente a servidor, entre otros.
3. El servidor ejecuta la tarea.
4. El servidor crea un mensaje de respuesta para el cliente en el que incluye el resultado de la tarea que este le pidió realizar.
5. El cliente recibe y desempaqueta el mensaje de respuesta del servidor. Continúa con su ejecución normal.[37]

Lo descrito anteriormente se representa en la figura 1:

[37] RUELAS, Uriel. ¿Qué es RPC (llamada a procedimiento remoto)?. Coding or not [en linea]. 19 de junio de 2018. [consultado: 29 de julio de 2018]. Disponible en internet: https://codingornot.com/que-es-rpc-llamada-a-procedimiento-remoto.

Figura 1 Funcionamiento de una llamada a procedimiento remoto



Fuente 1. RUELAS, Uriel. [Imagen]. ¿Qué es RPC (llamada a procedimiento remoto)?. 2018. [consultado: 29 de julio de 2018]. Disponible en internet: https://codingornot.com/que-es-rpc-llamada-a-procedimiento-remoto

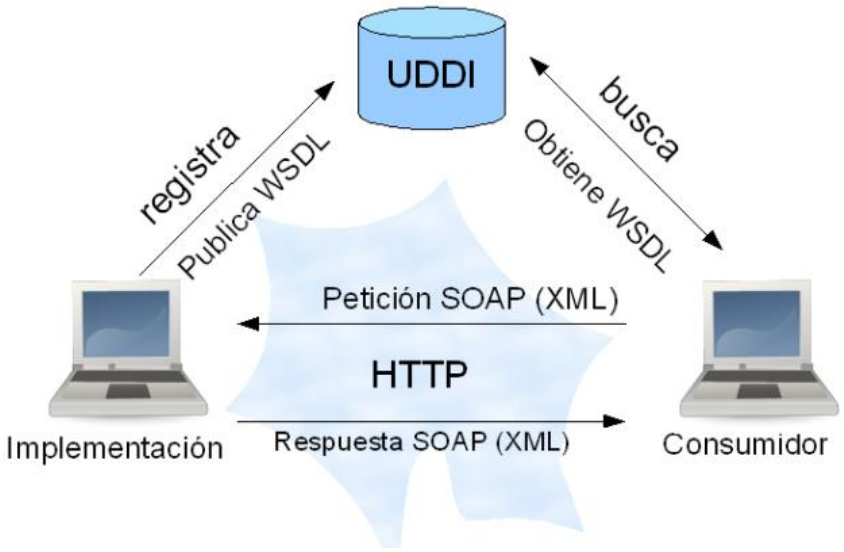
2.3.2.2.2 Simple Object Access Protocol (SOAP)

Protocolo que plantea la forma de comunicación entre las maquinas o aplicaciones. Los servicios web basados en este, intercambian datos por medio del estándar XML (Extensible Markup Language), el cual define la arquitectura y el formato de los mensajes. Existe una serie de tecnologías básicas que son fundamentales a la hora de desarrollar servicios web sobre el protocolo SOAP, que describen como se debe llevar a cabo la mensajería, el transporte de servicios, así como su descripción y localización. Como se dijo anteriormente la mensajería o el intercambio de datos se lleva a cabo gracias a XML el cual es responsable de establecer los mensajes de forma que puedan ser interpretados por cualquier aplicación; el transporte de servicios generalmente se lleva a cabo por medio del protocolo HTTP (Hypertext Transfer Protocol) aunque también se puede realizar mediante protocolos de transferencia de hipertexto como SMTP, FTP o BEEP; la descripción de servicios se hace mediante WSLD, que describe los requisitos funcionales necesarios para establecer la comunicación, este se basa en XML; por último para la localización de

los servicios se utiliza el protocolo UDDI (Universal Description, Discovery and Integration) el cual se puede definir como un directorio en el que se publican los servicios que están disponibles.

En la figura 2 se muestra el funcionamiento de las tecnologías descritas anteriormente

Figura 2 Funcionamiento de los servicios web SOAP



Fuente 2. LOS SANTOS, Alberto. Interacción a través de Servicios Web [imagen]. Metodologías para el Desarrollo de Servicios en la Web: Revisión de los servicios web SOAP/REST: Características y Rendimiento. Vigo.: Universidad de Vigo. 2009. 10 p. [consultado: 30 de julio de 2018]. Disponible en internet: http://www.albertolsa.com/wp-content/uploads/2009/07/mdsw-revision-de-los-servicios-web-soap_rest-alberto-los-santos.pdf

Existen desventajas en esta forma de implementar los servicios web, entre ellas, la complejidad al momento mostrar grandes cantidades de datos y la falta de escalabilidad de los sistemas. En resumen, un servicio web SOAP es una arquitectura orientada a servicios (SOA) que agrega ciertas características:

1. Excepto para datos binarios anexos, los mensajes deben ser transportados sobre SOAP.
2. La descripción de un servicio debe ser hecha en WSDL.
3. Uso de UDDI, que son las siglas del catálogo de negocios de Internet denominado Universal Description, Discovery and Integration. [38]

[38] LOS SANTOS, Alberto. Metodologías para el Desarrollo de Servicios en la Web: Revisión de los servicios web SOAP/REST: Características y Rendimiento [en línea]. Vigo.: Universidad de Vigo.

2.3.2.2.3 REpresentation State Transfer (REST)

Es un estilo que se basa en el protocolo HTTP y XML o JSON para realizar el intercambio de datos, se usa por lo general en sistemas distribuidos. “El término fue introducido en la tesis doctoral de Roy Fielding en 2000, quien es uno de los principales autores de la especificación de HTTP”[39]. Los servicios web basados en REST utilizan estándares para su funcionamiento tales como HTTP, URI, XML/JSON/HTML/GIF para representar los recursos y tipos MIME text/xml, text/json, text/html, y demás. Su construcción trae consigo ventajas como el tener una infraestructura ligera lo que hace que el desarrollo sea económico.

Algunas de las características más importantes de Rest son:

- Usa el protocolo HTTP (Hypertext Transfer Protocol) que permite la trasferencia de información, este estándar define la sintaxis que se deben utilizar para que se lleve a cabo la comunicación.
- No tiene estado, ya que se usa HTTP que es un protocolo sin estado. Esto implica que cada petición debe contener toda la información necesaria para la conexión puesto que el servidor no recuerda el estado de peticiones anteriores. Al realizar la petición se debe enviar la información de sesión, la cual se mantiene totalmente en el cliente, generalmente se realiza mediante el envío de un token.
- Para la representación de los recursos el lenguaje más usado es JSON, debido a que es mucho más ligero que otros, por lo tanto, requiere menos carga en su procesamiento

Como se dijo anteriormente Rest utiliza el protocolo HTTP, cuya especificación describe como se debe acceder a los recursos, para realizar este acceso hace uso de URIs, códigos de estado, métodos, cabeceras y contenidos. Las URIs permiten

2009. 9 p. [consultado: 30 de julio de 2018]. Disponible en internet: http://www.albertolsa.com/wp-content/uploads/2009/07/mdsw-revision-de-los-servicios-web-soap_rest-alberto-los-santos.pdf

[39] NAVARRO, Rafael. Rest vs web service. Modelado, Diseño e Implementación de Servicios Web [en línea]. 2006. 4 P. [consultado: en 30 de julio de 2018]. Disponible en internet: http://users.dsic.upv.es/~rnavarro/NewWeb/docs/RestVsWebServices.pdf

la identificación de los recursos, los cuales son los datos a los que se quiere acceder, estos pueden ser cualquier tipo de archivo. Existen dos tipos de URIs:

URL (Uniform Resource Locator): es la forma más utilizada y en resumidas cuentas es la dirección web donde se encuentra localizado el recurso. En el RFC 1738 se encuentra definida la estructura formal de la URL, en este explican la sintaxis y semántica que rige a este estándar.

URN (Uniform Resource Name): "un URN es una URI que identifica un recurso por su nombre en un espacio de nombres particular",[40] es decir, que identifica a un recurso, pero no indica el espacio en donde esté está localizado. En el RFC 8141 se define a profundidad este estándar.

Los códigos de estado HTTP señalan el resultado que se dio tras haber realizado una petición, están compuestos por tres dígitos, el primer digito corresponde al tipo de respuesta y los otros dos dígitos no especifican ninguna categoría. Existen cinco tipos de respuesta que son:

Respuestas informativas (1xx): este tipo de respuesta es provisional, por ejemplo, informa al navegador que la petición fue recibida y que aún no ha sido rechazada.

Respuestas exitosas (2xx): indican que la petición realizada por el navegador ha sido exitosa.

Redirecciones (3xx): indica al navegador que se deben realizar otras acciones para cumplir con la petición.

Errores del cliente (4xx): indica que ocurrió un error por parte del navegador.

Errores del servidor (5xx): El servidor señala que la petición puede que sea válida pero que ha ocurrió un error por culpa de este.

[40] NARVAEZ, Daniela y GONZALEZ, Sergio. Identificación de recursos web [en línea]. MDN web docs. (23 de mayo de 2018). [consultado: 2 de agosto de 2018]. Disponible en internet: https://developer.mozilla.org/es/docs/Web/HTTP/Basics_of_HTTP/Identificaci%C3%B3n_recursos_en_la_Web

En la sección 6 del RFC 2616 se listan los códigos de estado del estándar HTTP. Los métodos son un conjunto de operaciones bien definidas que se utilizan para especificar la operación que se va a realizar, estos también son conocidos como verbos HTTP, puesto que representan las acciones que se realizaran sobre cierto recurso. En la tabla 1 se pueden observar los verbos HTTP más importantes

Tabla 1. Métodos HTTP más importantes

Verbo	Descripción
GET	Este método sirve para solicitar datos, es decir, permite leer u obtener un recurso
POST	Se utiliza para realizar la actualización de los datos de un recurso en el servidor
PUT	Permite crear o agregar un recurso al servidor
DELETE	Este método elimina el recurso deseado

Adicionalmente existen otros verbos HTTP los cuales se pueden observar en la tabla 2

Tabla 2. Otros métodos HTTP

Verbo	Descripción
HEAD	Sirve para solicitar los metadatos de un recurso
CONNET	Permite acceder a una conexión del tipo TCP/IP
OPTIONS	Permite saber cuáles métodos HTTP soporta un recurso especifico
TRACE	Permite realizar pruebas a los mensajes que intercambian el cliente y el servidor
PATCH	Permite actualizar parcialmente un recurso

Por último, se tienen las cabeceras y el contenido HTTP, que conforman las peticiones realizadas al servidor. Las cabeceras o *headers* viajan con información

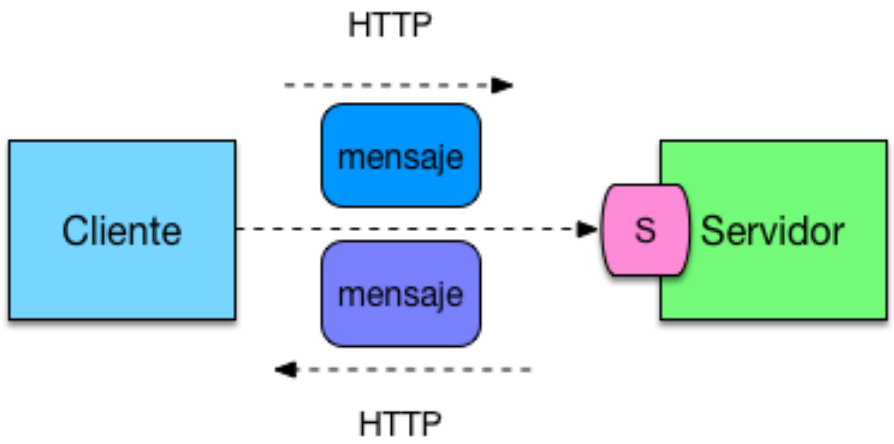
de la petición como métodos y/o credenciales de autenticación, control de la conexión, almacenamiento en la cache, tipo y/o tamaño del recurso, entre otros. Por otro lado, el contenido o *body* se refiere a la respuesta del servidor a la petición realizada por el navegador.

Este estilo de arquitectura aporta diferentes ventajas, entre ellas tenemos:

- Aumenta la escalabilidad
- Permite la independencia entre el cliente y el servidor
- Reducción de costos y bajo consumo de recurso
- Transfiere parte de la lógica al cliente
- Acoplamiento débil entre componentes
- Permite la utilización de diferentes tecnologías y lenguajes de programación

Resumiendo lo anteriormente dicho en la figura 3 se observa este estilo de arquitectura, en donde el cliente realiza una petición HTTP al servidor enviando un mensaje y el servidor de igual manera le responde a este.

Figura 3. Estilo de arquitectura REST



Fuente 3. ALVAREZ, Cecilio. [Imagen]. ¿ Que es REST ?. 2016. [consultado: 2 de agosto de 2018]. Disponible en internet: https://www.arquitecturajava.com/que-es-rest/

1.1.1.1 Empresas que usan servicios web

Son muchas las organizaciones que usan servicios web, entre ellas tenemos:

- Google LLC**: "**es una compañía principal subsidiaria de la multinacional estadounidense Alphabet Inc., cuya especialización son los productos y servicios relacionados con Internet, software, dispositivos electrónicos y otras tecnologías."[41] En la figura 4 se puede observar el logo de esta organización

Figura 4 Logo de Google

Fuente 4. WIKIPEDIA. [Imagen]. Archivo: Google 2015 logo.svg. [Consultado: 28 de julio de 2018]. Disponible en internet: https://es.m.wikipedia.org/wiki/Archivo:Google_2015_logo.svg

- Amazon: "es una compañía estadounidense de comercio electrónico y servicios de computación en la nube a todos los niveles con sede en la ciudad estadounidense de Seattle, Estado de Washington".[42] En la figura 5 se puede observar el logo de esta organización

Figura 5 Logo de Amazon

Fuente 5. LOGOS DE MARCAS. [Imagen]. Amazon logo. [Consultado: 28 de julio de 2018]. Disponible en internet: https://logosmarcas.com/amazon-logo/

- Movistar: "es la marca comercial de la multinacional española de telecomunicaciones Telefónica en España e Hispanoamérica, desde el 1 de mayo de 2010, para sus productos de telefonía fija, móvil, internet y

[41] WIKIPEDIA. Google. WIKIPEDIA La enciclopedia libre [en línea]. [Consultado: 28 de julio de 2018]. Disponible en internet: https://es.wikipedia.org/wiki/Google
[42] WIKIPEDIA. Amazon. WIKIPEDIA La enciclopedia libre [en línea]. [Consultado: 28 de julio de 2018]. Disponible en internet: https://es.wikipedia.org/wiki/Amazon

televisión. Puede hacer referencia a varias filiales:"[43] En la figura 6 se puede observar el logo de esta compañia

Figura 6 Logo de movistar

Fuente 6. EL GABO. [Imagen]. Movistar presenta su nueva identidad para este 2017. 2017. [Consultado: 28 de julio de 2018]. Disponible en internet: http://www.elpoderdelasideas.com/movistar-presenta-su-nueva-identidad-para-este-2017/

- Epic Games: esta empresa se dedica al desarrollo de videojuegos, son los creadores de Gears of War y de Fortnite. Se caracteriza por usar la tecnología Unreal Engine. En la figura 7 se puede observar el logo de esta organización

Figura 7 Logo de Epic Games

Fuente 7. Funko, Inc. [Imagen]. Funko and Epic Games Partner to Launch Fortnite™ Toys and Collectibles. 2018. [Consultado: 28 de julio de 2018]. Disponible en internet: https://globenewswire.com/news-release/2018/07/18/1538899/0/en/Funko-and-Epic-Games-Partner-to-Launch-Fortnite-Toys-and-Collectibles.html

- Cerner: Cerner Corporation se encarga de proveer soluciones tecnológicas, servicios, dispositivos y hardware de tecnología de información de salud. En la figura 8 se puede observar el logo de esta organización

[43] WIKIPEDIA. Movistar. WIKIPEDIA La enciclopedia libre [en línea]. [Consultado: 28 de julio de 2018]. Disponible en internet: https://es.wikipedia.org/wiki/Movistar

Figura 8 Logo de cerner

Fuente 8. CERNER. [Imagen]. Cardiac Risk. 2018. [Consultado: 28 de julio de 2018]. Disponible en internet: https://code.cerner.com/en/apps/ascvd-risk-calculator

- EBay: "es un marketplace destinado a comercio electrónico. Es uno de los pioneros en este tipo de transacciones, habiendo sido fundado en el año 1995"[44] En la figura 9 se puede observar el logo de esta organización

Figura 9 Logo de eBay

Fuente 9. WIKIPEDIA. [Imagen]. Archivo: EBay logo.svg. 2012. [Consultado: 28 de julio de 2018]. Disponible en internet: https://es.m.wikipedia.org/wiki/Archivo:EBay_logo.svg

- Outsystems: esta empresa se encarga de desarrollar aplicaciones web y móviles a diferentes organizaciones. Está ubicada en Atlanta, Georgia, Estados unidos. En la figura 10 se puede observar el logo de esta organización

Figura 10 Logo de outsystems

Fuente 10. OUTSYSTEMAS. [Imagen]. Mejore su negocio con OutSystems. 2018. [Consultado: 28 de julio de 2018]. Disponible en internet: https://www.totalebizsolutions.com/technology/outsystems/

- Coursera: "es una plataforma de educación virtual nacida en octubre de 2011 y desarrollada por académicos de la Universidad de Stanford con el fin de

[44] WIKIPEDIA. EBay. WIKIPEDIA La enciclopedia libre [en línea]. [Consultado: 28 de julio de 2018]. Disponible en internet: https://es.wikipedia.org/wiki/EBay

brindar oferta de educación masiva a la población"[45] En la figura 11 se puede observar el logo de esta organización

Figura 11 Logo de coursera

Fuente 11. WIKIPEDIA. [Imagen]. Archivo: Coursera logo.PNG. 2013. [Consultado: 28 de julio de 2018]. Disponible en internet: https://commons.wikimedia.org/wiki/File:Coursera_logo.PNG

- GrupoSantillana: "compañía global con sede en Tres Cantos (Comunidad de Madrid), antes Santillana Ediciones Generales, es un conjunto de editoriales dedicadas a la edición de libros de texto y contenidos educativos con presencia en España, 18 países de Iberoamérica, Portugal, Reino Unido y EE. UU"[46] En la figura 12 se puede observar el logo de esta organización

Figura 12 Logo del grupo Santillana

Fuente 12. LEIVA, Daniela. [Imagen]. Santillana. Blog Reconocido. Felicitaciones. 2016. [Consultado: 28 de julio de 2018]. Disponible en internet: http://trabajosdelcolegiodelourdes.blogspot.com/2016/06/santillana-blog-reconocido.html

2.3.3 Dashboard

Un tablero de mando o en ingles *Dashboard.* Es "un conjunto de indicadores esenciales organizados de manera tal que pueden reconocerse fácilmente patrones de rendimiento"[47]. Estos indicadores (KPI) deben ser los principales en una organización, y son representados por medio de una interfaz gráfica, de forma atractiva y sencilla, para que el usuario tenga la información que más le interesa

[45] WIKIPEDIA. Coursera. WIKIPEDIA La enciclopedia libre [en línea]. [Consultado: 28 de julio de 2018]. Disponible en internet: https://es.wikipedia.org/wiki/Coursera
[46] WIKIPEDIA. Grupo Santillana. WIKIPEDIA La enciclopedia libre [en línea]. [Consultado: 28 de julio de 2018]. Disponible en internet: https://es.wikipedia.org/wiki/Grupo_Santillana
[47] PUENTE, Raquel. Un tablero de mando para mercadeo [en línea]. En: Debates IESA. 2008, vol. XIII, no. 4, p 74-80. [Consultado: 10 de mayo de 2018]. Disponible en internet: http://servicios.iesa.edu.ve/portal/Articulos/14-Puente-Untablerodemando.pdf

rápidamente. Esta interfaz gráfica varía en cuanto a su diseño, formato y gráficos de una organización a otra, esto dependiendo de las necesidades de cada una de ellas. El objetivo principal de los tableros de mando es el de convertir los datos de una organización en información y esta a su vez en conocimiento para el logro de los objetivos de negocio. Es decir, los tableros de mando se encargan de mostrar de forma gráfica los principales indicadores de una empresa permitiendo así la toma acertada de decisiones.

A pesar que todos los *dashboards* son diferentes, deben cumplir las siguientes características básicas:

- Visualizar de manera amena, sencilla y entendible los datos, esto generalmente se realiza mediante gráficas. Además, debe ser mostrado en el mismo lenguaje del usuario.
- Mostrar los datos o indicadores más relevantes para el usuario (ni más, ni menos).
- Los datos mostrados deben ser objetivos, de tal modo, que proporcionen al usuario obtener su propio análisis.
- Todo se debe mostrar en una sola página, esto porque la idea de los *dashboards* es que se muestre de golpe la información
- Se utilizan más gráficos que textos
- Deben hablar el mismo lenguaje que el usuario final

2.3.3.1 Tipos de *dashboard*

Existen diversos tipos de dashboard, puesto que estos dependen del sector en el que se encuentre la empresa, así como de las áreas que la conforman. Algunos ejemplos de estos son:

- *Dashboard* para el área de recurso humano
- *Dashboard* para el área financiera
- *Dashboard* logístico
- *Dashboard* de ventas

- *Dashboard* de producción
- *Dashboard* de costos

2.3.3.2 Ventajas de los *dashboards*

- Disminuyen drásticamente la tediosa tarea de realizar y comprender informes manualmente.
- Facilitan la comprensión de los datos de una organización puesto que estos datos son mostrados por medio de graficas con diseños modernos e intuitivos.
- Ayudan a la toma de decisiones acertadas ya que se visualizan los indicadores más relevantes de la empresa de manera sencilla y amigable.
- Permiten la unificación de los datos relevantes de la organización.
- Convierten los datos en información y esta a su vez en conocimiento.
- Ayuda a ver posibles problemas que se estén presentado, se podría ver como una herramienta de diagnóstico.

2.3.4 Modelado de negocio

En el proceso de desarrollo de software se hace uso del modelado, el cual permite describir y representar elementos, aspectos y funcionalidades del sistema. El modelado se define entonces como “una herramienta conceptual que contiene un conjunto de objetos, conceptos y sus relaciones con el objetivo de expresar la lógica del negocio de una empresa”.[48]

El uso de un modelo ayuda a los desarrolladores de software a visualizar el sistema que se va a construir y como sus partes interactúan entre sí. Para lograr este objetivo se hace uso de un lenguaje de modelado, el cual permita representar las partes de un sistema de forma gráfica, a fin de reducir la complejidad a la hora de entender el software. Entre los lenguajes de modelado más conocidos se encuentran:

[48] DIAZ, Carolina. Modelos de negocio y medios online. Aproximación teórica a la cuestión. Revista digital Razón y Palabra [En línea], Marzo – Mayo 2013 [Consultado: 09 de Diciembre de 2018]. Disponible en Internet: http://www.razonypalabra.org.mx/N/N82/V82/39_Diaz_V82.pdf

- UML (*Unified Modeling Language*): lenguaje unificado de modelado, es un lenguaje grafico que permite representar las partes de un sistema de software, es usado en las etapas de desarrollo de software para especificar y describir sus métodos y procesos.
- BPMN (*Business Process Model and Notation*): notación para el modelado de procesos de negocio, "el objetivo de la notación para el modelado de procesos de negocios es proporcionar de una manera fácil de definir y analizar los procesos de negocios públicos y privados simulando un diagrama de flujo. La notación ha sido diseñada específicamente para coordinar la secuencia de los procesos y los mensajes que fluyen entre los participantes del mismo, con un conjunto de actividades relacionadas".[49]

A continuación, se va detallar un poco más a fondo el lenguaje de modelado unificado UML, ya que el desarrollo de este proyecto hace uso de algunas de sus gráficas y herramientas.

2.3.4.1 Diagramas UML

UML usa elementos gráficos generalizados y los asocia de manera que forma diagramas para representar aspectos estructurales y de comportamiento del sistema. En UML existen dos tipos de diagramas los estructurales que representan como se encuentra estructurado el sistema, y los de comportamiento, que representan la interacción entre los objetos del sistema.

Diagramas estructurales

- **Diagrama de clases:** muestra las clases de un sistema orientado a objetos junto con sus atributos, operaciones y relaciones.
- **Diagrama de componentes:** permite representar bloques de construcción de un sistema, compuesto por funcionalidades símiles.

[49] WIKIPEDIA. Ingeniería de software. WIKIPEDIA La enciclopedia libre [en línea]. [Consultado: 09 de Diciembre de 2018]. Disponible en internet: https://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software#cite_note-UML1-15

- **Diagrama de despliegue:** muestra la relación que hay entre el software y hardware del sistema.
- **Diagrama de objetos:** son similares a los diagramas de clases, muestran la interacción entre los objetos que componen el sistema.

Diagramas de comportamiento

- **Diagramas de secuencia:** muestra las interacciones entre los objetos que componen el sistema y el orden en que estas se efectúan.
- **Diagrama de casos de uso:** muestra la interacción de los actores con las funcionalidades del sistema, es el diagrama más usado de UML.
- **Diagramas de actividades:** Representa los flujos de trabajo de forma gráfica describiendo su interacción

Algo a tener en cuenta que cada uno de los diagramas se usa de forma distinta, pero cada uno de ellos se encuentra interrelacionados de alguna manera, a fin de estructurar y modelar cada aspecto de un software.

3 ANALISIS

La etapa de análisis dentro de la ingeniería del software, se encarga de obtener y describir cada uno de los requerimientos, que se capturan, gracias a que el cliente está involucrado en este proceso. La idea es formalizar la especificación del sistema mediante la construcción del modelo de análisis (en el numeral 3.2 de este mismo capítulo, se encuentra definido en que consiste este modelo de análisis). Los requerimientos se encargan de definir las características que el sistema debe cumplir. Al detallar las funcionalidades que se deben satisfacer, se está hablando de requerimientos funcionales. Una definición formal de esto, la brinda Bruegge y Dutoit en su libro Ingeniería de software orientado a objetos, que dice: "los requerimientos funcionales describen las interacciones entre el sistema y su ambiente, en forma independiente a su implementación."[50]

3.1 REQUERIMIENTOS FUNCIONALES

A continuación, se presenta el listado de los requerimientos funcionales capturados para el sistema

1. El sistema deberá permitir iniciar sesión mediante un usuario y una contraseña
2. El sistema deberá permitir al usuario crear uno o más *dashboards*, al realizar esto pedirá que ingrese un título.
3. El sistema deberá permitir que el usuario elimine sus *dashboards*
4. El sistema deberá permitir al usuario modificar sus *dashboards*
5. El sistema deberá permitir que el usuario visualice el listado de los *widgets* disponibles
6. El usuario puede agregar *widgets* a su *dashboard*
7. El usuario puede eliminar *widgets* de su *dashboard*
8. El usuario puede modificar las configuraciones del *widget*
9. El usuario puede buscar un *widget* especifico por medio del nombre de este

[50] BRUEGGE, Bernd y DUTOIT, Allen. Ingeniería de software orientado a objetos. Pearson educación.: México, 2002. 125 p.

10. El sistema deberá permitir que el usuario visualice la información concerniente a cada *widget*. Esta información incluye descripción del *widget*, calificaciones dadas, entre otras
11. El usuario podrá calificar los *widgets* que esté usando, podrá asignar una puntuación y agregar un comentario
12. El usuario podrá modificar las calificaciones que ha dado a los *widgets*. El sistema le debe permitir que modifique la puntuación y el comentario ingresado
13. El usuario podrá eliminar las calificaciones que realizó previamente a los *widgets*
14. El sistema debe permitir que el usuario opine sobre los comentarios que han realizado otros usuarios de los *widgets*.
15. El sistema debe permitir que el usuario elimine las opiniones realizadas a los comentarios de otros usuarios
16. El sistema debe permitir que el usuario modifique sus opiniones dadas a los comentarios de otros usuarios
17. El sistema deberá mostrar la información del usuario
18. El sistema deberá permitir que el usuario modifique su correo y contraseña
19. El sistema deberá permitir que el usuario cambie el tema por defecto del aplicativo

3.2 MODELO DEL SISTEMA

Como se mencionó al inicio de este capítulo, el producto de esta etapa, es un modelo de análisis, también llamado, modelo del sistema. La función de este, es lograr que el analista entienda las funcionalidades del sistema. Está comprendido por tres modelos: funcional, de datos y dinámico.

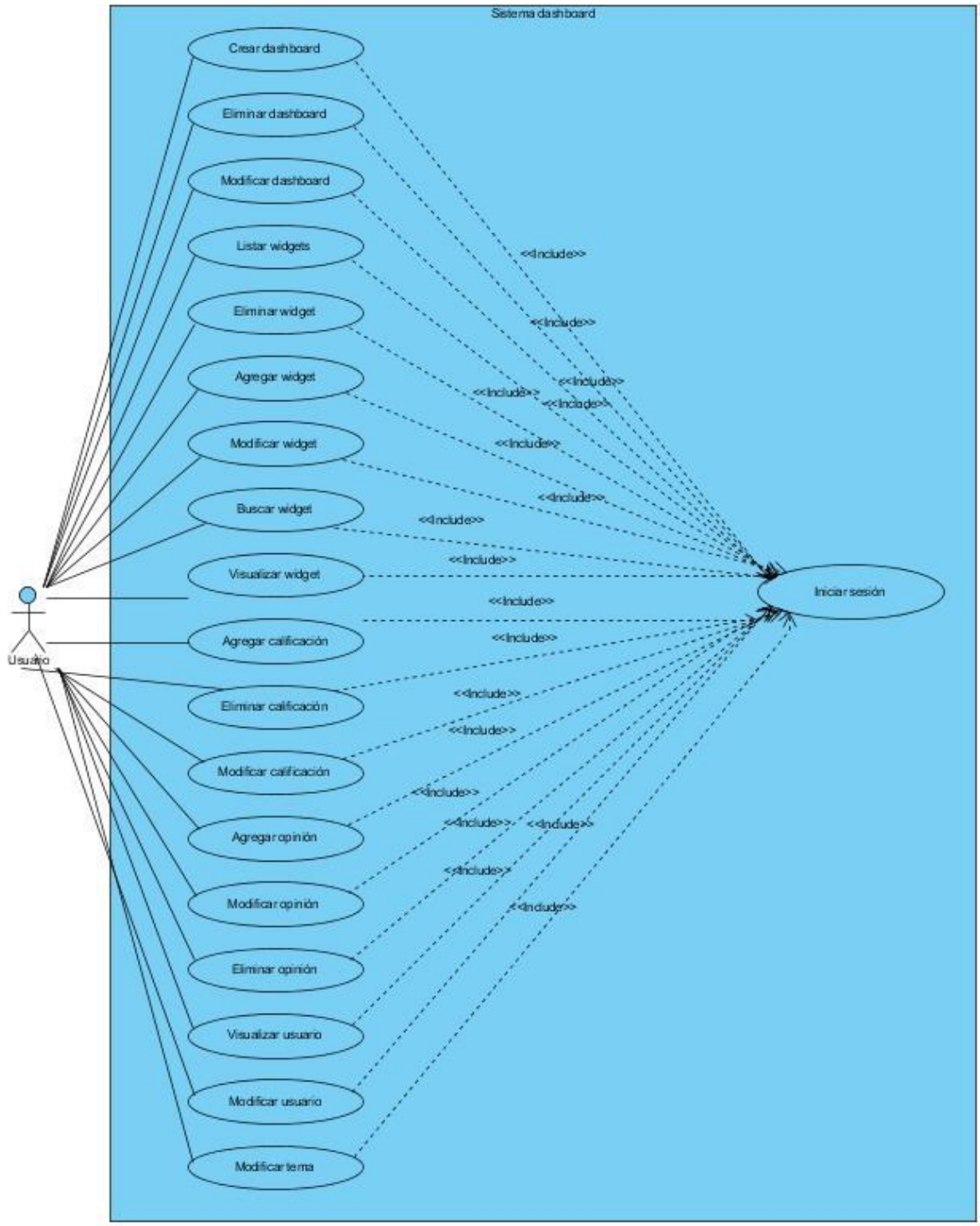
3.2.1 Modelo funcional

El modelo funcional sirve para describir la interacción que se da entre el usuario y el sistema. Esta interacción se representa mediante casos de uso, que permiten especificar esta comunicación. En esta sección, se realiza esto, presentando el diagrama de casos de uso, seguido por la especificación de cada uno de ellos.

3.2.1.1 Casos de uso

A continuación, en la figura 13 se muestra mediante el diagrama de casos de uso las relaciones entre el actor, los casos de uso y el sistema. En este caso y para fines de este proyecto se cuenta con un actor, que es el usuario y con diecinueve casos de uso que corresponden a los requerimientos funcionales del sistema, descritos en el numeral 3.1, de este mismo capítulo. El caso de uso iniciar sesión, es incluido en los demás, esto implica que sin este los otros no pueden funcionar, es decir, si el usuario no inicia sesión, no podrá acceder a las funcionalidades que brinda el sistema.

Figura 13. Diagrama de casos de uso



3.2.1.2 Especificación de los casos de uso

Para una mayor comprensión del diagrama anterior se especifica cada uno de los casos de uso, teniendo en cuenta su flujo principal, así como los alternativos.

3.2.1.2.1 CU1. Iniciar sesión

Tabla 3. Flujo principal del caso de uso 1: iniciar sesión

Caso de uso	CU1. Iniciar sesión		
Descripción	El sistema debe permitir que el usuario inicie sesión ingresando un usuario y una contraseña, seguido de esto debe mostrar la interfaz correspondiente		
Precondición			
Postcondición	El sistema cargara la interfaz de usuario		
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime el botón transacciones	
	2		Muestra un formulario donde podrá ingresar el usuario
	3	Ingresa el usuario	
	4	Envía el usuario	
	5		Muestra un formulario donde podrá ingresar la contraseña
	6	Ingresa la contraseña	
	7	Oprime ingresar	
	8		Muestra la interfaz correspondiente al usuario

Tabla 4. Flujo alternativo 1 del caso de uso 1: iniciar sesión

Precondición			
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime el botón transacciones	
	2		Muestra un formulario donde podrá ingresar el usuario
	3	Envía el usuario	
	4		Muestra un mensaje indicando que el campo usuario es obligatorio

Tabla 5. Flujo alternativo 2 del caso de uso 1: iniciar sesión

Precondición			
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime el botón transacciones	
	2		Muestra un formulario donde podrá ingresar el usuario
	3	Ingresa el usuario	
	4	Envía el usuario	
	5		Muestra un formulario donde podrá ingresar la contraseña
	6	Oprime ingresar	
	7		Muestra un mensaje indicando que el campo contraseña es obligatorio

Tabla 6. Flujo alternativo 3 del caso de uso 1: iniciar sesión

Precondición			
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime el botón transacciones	
	2		Muestra un formulario donde podrá ingresar el usuario
	3	Ingresa el usuario	
	4	Oprime siguiente	
	5		Muestra un formulario donde podrá ingresar la contraseña
	6	Ingresa la contraseña	
	7	Oprime ingresar	
	8		Muestra un mensaje indicando que los campos usuario o contraseña son incorrectos

3.2.1.2.2 CU2. Crear *dashboard*

Tabla 7. Flujo principal del caso de uso 2: crear dashboard

Caso de uso	CU2. Crear dashboard		
Descripción	El usuario puede crear uno o más dashboards, al realizar esto el sistema debe pedir el título de cada uno de ellos		
Precondición	Haber iniciado sesión		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime crear Dashboard	
	2		Muestra un formulario en donde podrá ingresar el título del Dashboard
	3	Ingresa el titulo	
	4	Oprime aceptar	
	5		Muestra una ventana con el *dashboard* creado

Tabla 8. Flujo alternativo 1 del caso de uso 2: crear dashboard

Precondición	Haber iniciado sesión		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime crear dashboard	
	2		Muestra un formulario en donde podrá ingresar el título del dashboard
	3	Ingresa el titulo	
	4	Oprime cancelar	

Tabla 9. Flujo alternativo 2 del caso de uso 2: crear dashboard

Precondición	Haber iniciado sesión		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime crear dashboard	
	2		Muestra un formulario en donde podrá ingresar el título del dashboard
	3	Oprime aceptar	
	4		Muestra un mensaje donde dice que el campo título es obligatorio

3.2.1.2.3 CU3. Eliminar *dashboard*

Tabla 10. Flujo principal del caso de uso 3: eliminar dashboard

Caso de uso	CU3. Eliminar dashboard		
Descripción	El usuario puede eliminar el o los dashboards que desee		
Precondición	Haber iniciado sesión, tener dashboards creados		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime eliminar Dashboard	
	2		Muestra una ventana pidiendo confirmación para eliminar el Dashboard
	3	Oprime aceptar	

Tabla 11. Flujo alternativo del caso de uso 3: eliminar dashboard

Precondición	Haber iniciado sesión, tener dashboards creados		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime eliminar Dashboard	
	2		Muestra una ventana pidiendo confirmación para eliminar el Dashboard
	3	Oprime cancelar	

3.2.1.2.4 CU4. Modificar *dashboard*

Tabla 12. Flujo principal del caso de uso 4: modificar dashboard

Caso de uso	CU4. Modificar dashboard		
Descripción	El sistema debe permitir que el usuario modifique sus dashboards		
Precondición	Haber iniciado sesión, tener dashboards creados		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime modificar Dashboard	
	2		Muestra un formulario en donde podrá modificar el titulo
	3	Ingresa el nuevo titulo	
	4	Oprime guardar	
	5		Muestra el Dashboard con los cambios realizados

Tabla 13. Flujo alternativo 1 del caso de uso 4: modificar dashboard

Precondición	Haber iniciado sesión, tener dashboards creados		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime modificar Dashboard	
	2		Muestra un formulario donde podrá modificar el titulo
	3	Ingresa el nuevo titulo	
	4	Oprime cancelar	
	5		Muestra el Dashboard sin cambios

Tabla 14. Flujo alternativo 2 del caso de uso 4: modificar dashboard

Precondición	Haber iniciado sesión, tener dashboards creados		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime modificar Dashboard	
	2		Muestra un formulario en donde podrá modificar el titulo
	3	Oprime aceptar	
	4		Muestra el Dashboard sin cambios

3.2.1.2.5 CU5. Listar *widgets*

Tabla 15. Flujo principal del caso de uso 5: listar widgets

Caso de uso	CU5. Listar *widgets*		
Descripción	El sistema debe permitir que el usuario visualice el listado de los *widgets* disponibles		
Precondición	Haber iniciado sesión		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime la opción *widgets*	
	2		Muestra una ventana con los *widgets* disponibles

3.2.1.2.6 CU6. Agregar *widget*

Tabla 16. Flujo principal del caso de uso 6: agregar widget

Caso de uso	CU6. Agregar *widget*		
Descripción	El sistema permite que el usuario agregue widgets a sus dashboards		
Precondición	Haber iniciado sesión, tener por lo menos un *dashboard* creado		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime agregar *widget*	
	2		Muestra un formulario de las configuraciones del *widget*

	3	Ingresa las configuraciones del *widget*	
	4	Oprime aceptar	
	5		Muestra el d*ashboard* con el nuevo *widget* agregado

Tabla 17. Flujo alternativo 1 del caso de uso 6: agregar widget

Precondición	Haber iniciado sesión, tener por lo menos un *dashboard* creado		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime agregar widget	
	2		Muestra un formulario de las configuraciones del widget
	3	Oprime aceptar	
	4		Muestra un mensaje donde indica que la configuración es obligatoria

Tabla 18. Flujo alternativo 2 del caso de uso 6: agregar widget

Precondición	Haber iniciado sesión, tener por lo menos un dashboard creado		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime agregar widget	

	2		Muestra un formulario de las configuraciones del widget
	3	Ingresa las configuraciones del widget	
	4	Oprime cancelar	

3.2.1.2.7 CU7. Eliminar *widget*

Tabla 19. Flujo principal del caso de uso 7: eliminar widget

Caso de uso	CU7. Eliminar widget		
Descripción	El usuario puede eliminar los widgets de sus dashboards según como desee		
Precondición	Haber iniciado sesión, tener por lo menos un widget agregado		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime eliminar widget del dashboard	
	2		Muestra una ventana de confirmación para eliminar el widget
	3	Oprime aceptar	

Tabla 20. Flujo alternativo del caso de uso 7: eliminar widget

Precondición	Haber iniciado sesión, tener por lo menos un widget agregado		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime eliminar widget del Dashboard	

	2		Muestra una ventana de confirmación para eliminar el widget
	3	Oprime cancelar	
	4		Muestra ventana con el widget

3.2.1.2.8 CU8. Modificar *widget*

Tabla 21. Flujo principal del caso de uso 8: modificar widget

Caso de uso	CU8. Modificar widget		
Descripción	El sistema debe permitir que el usuario modifique los widgets que tiene agregados		
Precondición	Haber iniciado sesión, tener por lo menos un widget agregado		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime personalizar widget	
	2		Muestra un formulario de las configuraciones del widget
	3	Ingresa las nuevas configuraciones	
	4	Oprime aceptar	
	5		Muestra el widget con las nuevas configuraciones

Tabla 22. Flujo alternativo 1 del caso de uso 8: modificar widget

Precondición	Haber iniciado sesión, tener por lo menos un widget agregado		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime personalizar widget	
	2		Muestra un formulario de las configuraciones del widget
	3	Ingresa las nuevas configuraciones	
	4	Oprime cancelar	
	5		Muestra el widget sin cambios

Tabla 23. Flujo alternativo 2 del caso de uso 8: modificar widget

Precondición	Haber iniciado sesión, tener por lo menos un widget agregado		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime personalizar widget	
	2		Muestra un formulario de las configuraciones del widget
	3	Oprime aceptar	
	4		Muestra el widget sin cambios

3.2.1.2.9 CU9. Buscar *widget*

Tabla 24. Flujo principal del caso de uso 9: buscar widget

Caso de uso	CU9. Buscar widget		
Descripción	El usuario puede buscar los widgets por medio de su nombre		
Precondición	Haber iniciado sesión		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime buscar widget	
	2		Muestra un formulario para buscar el widget
	3	Ingresa el nombre del widget que desea buscar	
	4	Oprime buscar	
	5		Muestra una ventana con los widgets que coinciden con la búsqueda

Tabla 25. Flujo alternativo del caso de uso 9: buscar widget

Precondición	Haber iniciado sesión		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime buscar widget	
	2		Muestra un formulario para buscar el widget
	3	Oprime buscar	
	4		Muestra un mensaje donde indica que el campo es obligatorio

3.2.1.2.10 CU10. Visualizar *widget*

Tabla 26. Flujo principal del caso de uso 10: visualizar widget

Caso de uso	CU10. Visualizar widget		
Descripción	El sistema debe permitir que el usuario visualice la información concerniente a cada widget. Esta información incluye descripción del widget, calificaciones dadas, entre otras		
Precondición	Haber iniciado sesión		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime ver información	
	2		Muestra una ventana con la información del widget, esta incluye descripción, vista previa, calificaciones realizadas por otros usuarios y estadísticas de las calificaciones

3.2.1.2.11 CU11. Agregar calificación

Tabla 27. Flujo principal del caso de uso 11: agregar calificación

Caso de uso	CU11. Agregar calificación		
Descripción	El usuario puede calificar los widgets que tenga agregados en sus dashboards		
Precondición	Haber iniciado sesión, tener agregado el widget por lo menos una vez		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime calificar widget	

	2		Muestra un formulario donde podrá seleccionar la puntuación que le da al widget
	3	Selecciona la puntuación deseada	
	4	Oprime enviar	
	5		Muestra un formulario donde podrá ingresar un comentario
	6	Ingresa el comentario	
	7	Oprime enviar	
	8		Muestra una ventana con la calificación agregada

Tabla 28. Flujo alternativo 1 del caso de uso 11: agregar calificación

Precondición	Haber iniciado sesión, tener agregado el widget por lo menos una vez		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime calificar widget	
	2		Muestra un formulario donde podrá seleccionar la puntuación que le da al widget
	3	Selecciona la puntuación deseada	
	4	Oprime enviar	
	5		Muestra un formulario donde podrá ingresar un comentario

	6	Oprime enviar	
	7		Muestra una ventana con la calificación agregada

Tabla 29. Flujo alternativo 2 del caso de uso 11: agregar calificación

Precondición	Haber iniciado sesión, tener agregado el widget por lo menos una vez		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime calificar widget	
	2		Muestra un formulario donde podrá seleccionar la puntuación que le da al widget
	3	Selecciona la puntuación deseada	
	4	Oprime enviar	
	5		Muestra un formulario donde podrá ingresar un comentario
	6	Ingresa el comentario	
	7	Oprime cancelar	

Tabla 30. Flujo alternativo 3 del caso de uso 11: agregar calificación

Precondición	Haber iniciado sesión, tener agregado el widget por lo menos una vez		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime calificar widget	

	2		Muestra un formulario donde podrá seleccionar la puntuación que le da al widget
	3	Selecciona la puntuación deseada	
	4	Oprime cancelar	

Tabla 31. Flujo alternativo 4 del caso de uso 11: agregar calificación

Precondición	Haber iniciado sesión, tener agregado el widget por lo menos una vez		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime calificar widget	
	2		Muestra un formulario donde podrá seleccionar la puntuación que le da al widget
	3	Oprime enviar	
	4		Muestra un mensaje indicando que el campo puntuación es obligatorio

3.2.1.2.12 CU12. Modificar calificación

Tabla 32. Flujo principal del caso de uso 12: modificar calificación

Caso de uso	CU12. Modificar calificación
Descripción	El usuario podrá modificar las calificaciones que le ha dado a los widgets
Precondición	Haber calificado el widget

Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime editar calificación	
	2		Muestra un formulario donde podrá modificar la puntuación
	3	Selección la nueva puntuación	
	4	Oprime enviar	
	5		Muestra un formulario donde podrá modificar el comentario
	6	Ingresa el nuevo comentario	
	7	Oprime enviar	
	8		Muestra una ventana con la calificación actualizada

Tabla 33. Flujo alternativo 1 del caso de uso 12: modificar calificación

Precondición	Haber calificado el widget		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime editar calificación	
	2		Muestra un formulario donde podrá modificar la puntuación
	3	Oprime enviar	
	4		Muestra un formulario donde podrá modificar el comentario
	5	Oprime enviar	

Tabla 34. Flujo alternativo 2 del caso de uso 12: modificar calificación

Precondición	Haber calificado el widget		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime editar calificación	
	2		Muestra un formulario donde podrá modificar la puntuación
	3	Selecciona la nueva puntuación	
	4	Oprime enviar	
	5		Muestra un formulario donde podrá modificar el comentario
	6	Oprime enviar	
	7		Muestra una ventana con la calificación actualizada

Tabla 35. Flujo alternativo 3 del caso de uso 12: modificar calificación

Precondición	Haber calificado el widget		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime editar calificación	
	2		Muestra un formulario donde podrá modificar la puntuación
	3	Oprime enviar	
	4		Muestra un formulario donde podrá modificar el comentario

	5	Ingresa el nuevo comentario	
	6	Oprime enviar	
	7		Muestra una ventana con la calificación actualizada

Tabla 36. Flujo alternativo 4 del caso de uso 12: modificar calificación

Precondición	Haber calificado el widget		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime editar calificación	
	2		Muestra un formulario donde podrá modificar la puntuación
	3	Oprime cancelar	

3.2.1.2.13 CU13. Eliminar calificación

Tabla 37. Flujo principal del caso de uso 13: eliminar calificación

Caso de uso	CU13. Eliminar calificación		
Descripción	El usuario podrá eliminar las calificaciones que realizo a los widgets		
Precondición	Haber iniciado sesión, haber calificado el widget		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime eliminar calificación	
	2		Muestra una ventana con la opción de calificar widget

3.2.1.2.14 CU14. Agregar opinión

Tabla 38. Flujo principal del caso de uso 14: agregar opinión

Caso de uso	CU14. Agregar opinión		
Descripción	El sistema debe permitir que el usuario opine sobre los comentarios que han realizado otros usuarios de los widgets		
Precondición	El widget debe tener comentarios agregados		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime una de las opciones de agregar opinión	
	2		Muestra el comentario con la opinión agregada

3.2.1.2.15 CU15. Eliminar opinión

Tabla 39. Flujo principal del caso de uso 15: Eliminar opinión

Caso de uso	CU15. Eliminar opinión		
Descripción	El sistema debe permitir que el usuario elimine las opiniones realizadas a los comentarios		
Precondición	El comentario debe tener agregada la opinión		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime la opción que esta agregada, esta será la que se va a eliminar	
	2		Muestra el comentario sin la opinión

3.2.1.2.16 CU16. Modificar opinión

Tabla 40. Flujo principal del caso de uso 16: Modificar opinión

Caso de uso	CU16. Modificar opinión		
Descripción	El sistema debe permitir que el usuario modifique sus opiniones dadas a los comentarios de otros usuarios		
Precondición	El comentario debe tener opiniones previamente agregadas		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime una opción diferente de la previamente seleccionada al agregar la opinión	
	2		Muestra el comentario con la opinión modificada

3.2.1.2.17 CU17. Visualizar usuario

Tabla 41. Flujo principal del caso de uso 17: visualizar usuario

Caso de uso	CU17. Visualizar usuario		
Descripción	El sistema debe permitir que el usuario visualice su información		
Precondición	Haber iniciado sesión		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime ver perfil	
	2		Muestra una ventana con la información del usuario

3.2.1.2.18 CU18. Modificar usuario

Tabla 42. Flujo principal del caso de uso 18: modificar usuario

Caso de uso	CU20. Modificar usuario		
Descripción	El usuario tendrá la opción de modificar su correo y su contraseña		
Precondición	Haber iniciado sesión		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime modificar usuario	
	2		Muestra un formulario donde podrá modificar el correo electrónico
	3	Ingresa el nuevo correo electrónico	
	4	Oprime enviar	
	5		Muestra un formulario donde podrá modificar la contraseña
	6	Ingresa la nueva contraseña	
	7	Oprime enviar	
	8		Muestra una ventana con el usuario modificad

Tabla 43. Flujo alternativo 1 del caso de uso 18: modificar usuario

Precondición	Haber iniciado sesión		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime modificar usuario	

	2		Muestra un formulario donde podrá modificar el correo electrónico
	3	Ingresa el nuevo correo electrónico	
	4	Oprime enviar	
	5		Muestra un formulario donde podrá modificar la contraseña
	6	Oprime enviar	
	7		Muestra una ventana con el usuario modificado

Tabla 44. Flujo alternativo 2 del caso de uso 18: modificar usuario

Precondición	Haber iniciado sesión		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime modificar usuario	
	2		Muestra un formulario donde podrá modificar el correo electrónico
	3	Oprime enviar	
	4		Muestra un formulario donde podrá modificar la contraseña
	5	Ingresa la nueva contraseña	
	6	Oprime enviar	
	7		Muestra una ventana con el usuario modificado

Tabla 45. Flujo alternativo 3 del caso de uso 18: modificar usuario

Precondición	Haber iniciado sesión		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime modificar usuario	
	2		Muestra un formulario donde podrá modificar el correo electrónico
	3	Oprime enviar	
	4		Muestra un formulario donde podrá modificar la contraseña
	5	Oprime enviar	
	6		Muestra la ventana del usuario

3.2.1.2.19 CU19. Modificar tema

Tabla 46. Flujo principal del caso de uso 19: modificar tema

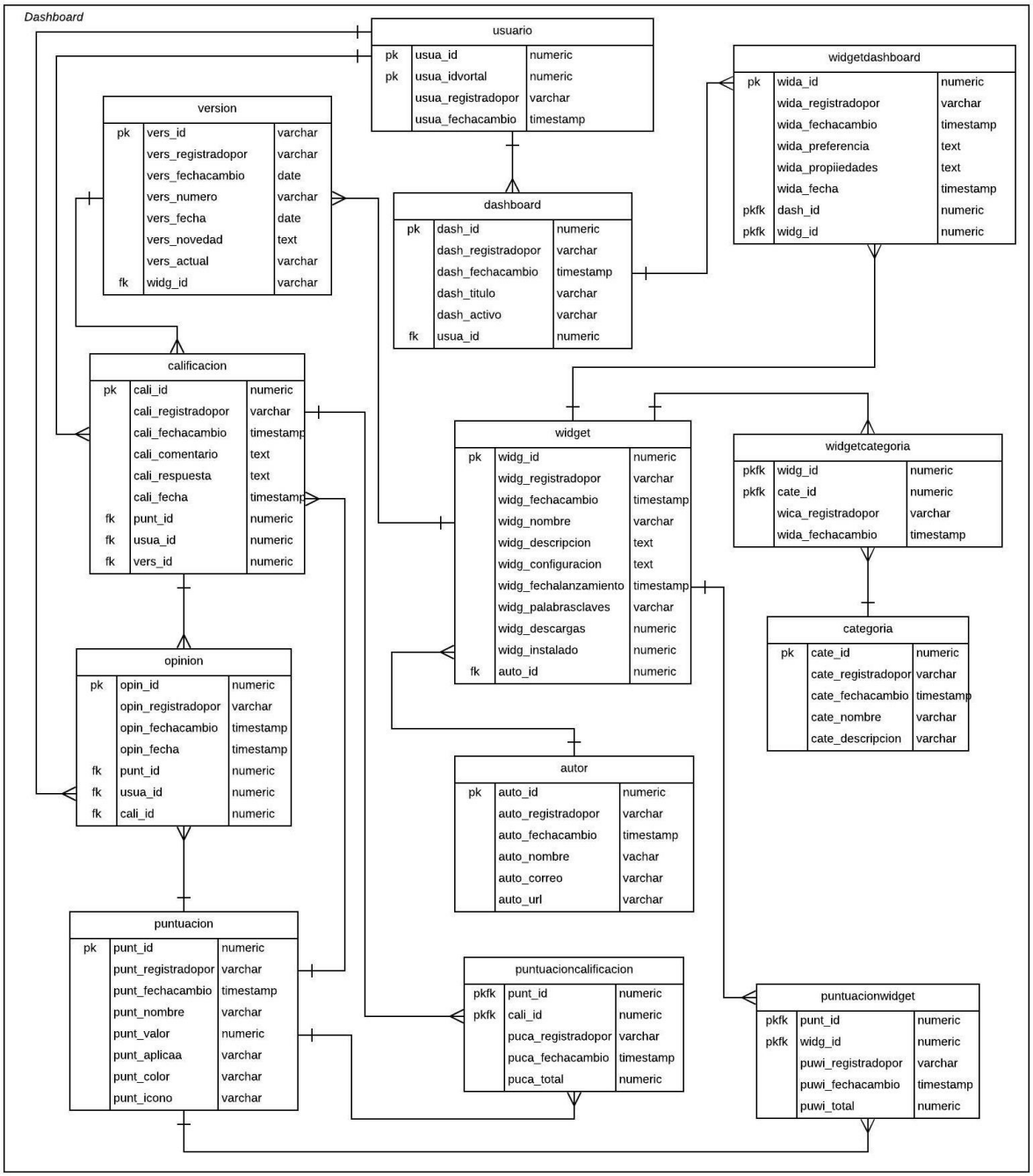
Caso de uso	CU19. Modificar tema		
Descripción	El sistema deberá permitir que el usuario cambie el tema por defecto del aplicativo		
Precondición	Haber iniciado sesión		
Postcondición			
Flujo de eventos		**Usuario**	**Sistema**
	1	Oprime el botón modificar tema	
	2		Muestra el listado de los temas disponibles
	3	Selecciona el nuevo tema	

4	Cambia el tema de la aplicación

3.2.2 Modelo de datos

El modelo de datos se encarga de definir la estructura de los datos persistentes en el sistema. En este caso se representa mediante un diagrama entidad relación, que está reflejado en la figura 14.

Figura 14. Diagrama entidad relación del sistema



3.2.2.1 Diccionario de datos

Tabla 47. Tabla dashboard.usuario

Nombre de la tabla	usuario				
Descripción	Relaciona al usuario del vortal para tener la referencia que el mismo ya ingreso en el dashboard				
Relaciones					
Campo	**Tipo**	**Tamaño**	**Descripción**	**Pk**	**Fk**
usua_id	Numeric	30	Identificador del usuario dentro del dashboard	Si	
usua_idvortal	Numeric	30	Almacena el id del usuario relacionado en la tabla general.usuario del vortal	Si	
usua_registradopor	Varchar	250	Campo de auditoria		
usua_fechacambio	Timestamp		Campo de auditoria		

Tabla 48. Tabla dashboard.dashboard

Nombre de la tabla	dashboard				
Descripción	Guarda los dashboard creados por el usuario				
Relaciones	Foráneo de la tabla dashboard.usuario				
Campo	**Tipo**	**Tamaño**	**Descripción**	**Pk**	**Fk**
dash_id	Numeric	30	Almacena el identificador del dashboard	Si	

dash_registradopor	Varchar	250	Campo de auditoria		
dash_fechacambio	Timestamp		Campo de auditoria		
dash_titulo	Varchar	250	Almacena el titulo asignado para el dashboard		
dash_activo	Varchar	1	Identifica si el dashboard actual está activo o no		
usua_id	Numeric	30	Identificador del usuario que creo el dashboard		Si

Tabla 49. Tabla dashboard.autor

Nombre de la tabla	autor				
Descripción	Guarda la relación de los autores de los widget				
Relaciones					
Campo	**Tipo**	**Tamaño**	**Descripción**	**Pk**	**Fk**
auto_id	Numeric	30	Almacena el identificador del autor	Si	
auto_registradopor	Varchar	250	Campo de auditoria		
auto_fechacambio	Timestamp		Campo de auditoria		
auto_nombre	Varchar	250	Almacena el nombre del autor que crea un widget		

auto_correo	Varchar	250	Almacena el correo del autor del widget para su posterior contacto		
auto_url	Varchar	1000	Almacena la dirección del sitio Web del autor del widget		

Tabla 50. Tabla dashboard.widget

Nombre de la tabla	widget				
Descripción	Relaciona los widgets disponibles				
Relaciones	Foránea de la tabla dashboard.autor				
Campo	**Tipo**	**Tamaño**	**Descripción**	**Pk**	**Fk**
widg_id	Numeric	30	Almacena el identificador del widget	Si	
widg_registradopor	Varchar	250	Campo de auditoria		
widg_fechacambio	Timestamp		Campo de auditoria		
widg_nombre	Varchar	250	Nombre para identificar el widget		
widg_descripcion	Text		Breve descripción de lo que hace el widget		

widg_configuracion	Text		Almacena en formato JSON la información de configuración del widget		
widg_fechalanzamiento	Timestamp		Almacena la fecha de cuando fue liberado por primera vez el widget		
widg_palabrasclaves	Varchar	500	Relaciona las palabras claves asociadas a este widget separadas por coma (,)		
widg_descargas	Numeric		Mantiene la relación de cuantas veces ha sido instalado el widget en el tiempo		
widg_instalado	Numeric		Mantiene la relación de cuantas veces está instalado el widget en estos momentos		
auto_id	Numeric	30	Identificador del autor		Si

Tabla 51. Tabla dashboard.widgetdashboard

Nombre de la tabla		widgetdashboard			
Descripción		Contiene la relación de los widgets en cada dashboard			
Relaciones		Foráneo de la tabla dashboard.dashboard Foráneo de la tabla dashboard.widget			
Campo	**Tipo**	**Tamaño**	**Descripción**	**Pk**	**Fk**
wida_id	Numeric	30	Almacena el identificador de la relación del widget en el dashboard	Si	
wida_registradopor	Varchar	250	Campo de auditoria		
wida_fechacambio	Timestamp		Campo de auditoria		
wida_preferencia	Text		Almacena las preferencias del usuario usadas para el pintado del widget en formato JSON		
wida_propiedades	Text		Almacena las propiedades definidas por el usuario para este widget en el dashboard en formato JSON		
wida_fecha	Timestamp		Almacena la fecha en que fue asignado		

			el widget al dashboard		
dash_id	Numeric	30	Identificador del dashboard		Si
widg_id	Numeric	30	Identificador del widget		Si

Tabla 52.Tabla dashboard.categoriat

Nombre de la tabla	categoria				
Descripción	Relaciona las categorías de agrupación de los widgets				
Relaciones					
Campo	**Tipo**	**Tamaño**	**Descripción**	**Pk**	**Fk**
cate_id	Numeric	30	Almacena el identificador de una categoría	Si	
cate_registradopor	Varchar	250	Campo de auditoria		
cate_fechacambio	Timestamp		Campo de auditoria		
cate_nombre	Varchar	250	Nombre para identificar la categoría		
cate_descripcion	Varchar	500	Breve descripción de lo que representa la categoría		

Tabla 53. Tabla dashboard.widgetcategoria

Nombre de la tabla		widgetcategoria			
Descripción		Relaciona los widgets disponibles para adicionar en cada dashboard			
Relaciones		Foráneo de la tabla dashboard.widget Foráneo de la tabla dashboard.categoria			
Campo	**Tipo**	**Tamaño**	**Descripción**	**Pk**	**Fk**
widg_id	Numeric	30	Identificador del widget	Si	Si
cate_id	Numeric	30	Identificador de la categoría	Si	Si
wica_registradopor	Varchar	250	Campo de auditoria		
wica_fechacambio	Timestamp		Campo de auditoria		

Tabla 54. Tabla dashboard.puntuacion

Nombre de la tabla		puntuación			
Descripción		Relaciona las puntuaciones usadas para el manejo de estadísticas y calificación de los widget y calificaciones			
Relaciones					
Campo	**Tipo**	**Tamaño**	**Descripción**	**Pk**	**Fk**
punt_id	Numeric	30	Almacena el identificador de la puntuación	Si	
punt_registradopor	Varchar	250	Campo de auditoria		
punt_fechacambio	Timestamp		Campo de auditoria		

punt_nombre	Varchar	250	Nombre para la puntuación		
punt_valor	Numeric	4	Valor representativo de la puntuación usado para las estadísticas		
punt_aplicaa	Varchar	1	Identifica a qué tipo de evaluación aplica, (w=widget, c=calificación)		
punt_color	Varchar	10	Representa el color para el pintado de la puntuación en los gráficos estadísticos		
punt_icono	Varchar	20	Nombre del icono para pintar teniendo en cuenta el estilo de material		

Tabla 55. Tabla dashboard.version

Nombre de la tabla		versión			
Descripción		Guarda la relación de las versiones que ha tenido un widget			
Relaciones		Extranjero de la tabla dashboard.widget			
Campo	**Tipo**	**Tamaño**	**Descripción**	**Pk**	**Fk**
vers_id	Numeric	30	Almacena el identificador de la versión	Si	

vers_registradopor	Varchar	250	Campo de auditoria		
vers_fechacambio	Timestamp		Campo de auditoria		
vers_numero	Varchar	250	Guarda el número de la versión del widget		
vers_fecha	Timestamp		Almacena la fecha en la que se subió la versión		
vers_novedad	Text		Almacena los cambios hechos en la versión		
vers_actual	Varchar	1	Identifica cual es la versión actual de un widget (1 = actual, 0 = antigua)		
widg_id	Numeric	30	Identificador del widget		Si

Tabla 56. Tabla dashboard.puntuacionwidget

Nombre de la tabla	puntuacionwidget				
Descripción	Guarda la relación de las puntuaciones dadas al widget de manera estadística				
Relaciones	Foráneo de la tabla dashboard.puntuacion Foráneo de la tabla dashboard.widget				
Campo	**Tipo**	**Tamaño**	**Descripción**	**Pk**	**Fk**

punt_id	Numeric	30	Almacena el identificador de la puntuación	Si	Si
widg_id	Numeric	30	Almacena el identificador del widget	Si	Si
puwi_registradopor	Varchar	250	Campo de auditoria		
puwi_fechacambio	Timestamp		Campo de auditoria		
puwi_total	Numeric	10	Almacena el total de puntuaciones sobre el widget, cuantas 1 estrellas, cuantas 2 estrellas …		

Tabla 57. Tabla dashboard.calificacion

Nombre de la tabla	calificacion				
Descripción	Guarda cada calificacion dada a un widget				
Relaciones	Foráneo de la tabla dashboard.puntuacion Foráneo de la tabla dashboard.usuario Foráneo de la tabla dashboard.version				
Campo	**Tipo**	**Tamaño**	**Descripción**	**Pk**	**Fk**
cali_id	Numeric	30	Almacena el identificador de la calificación	Si	
cali_registradopor	Varchar	250	Campo de auditoria		
cali_fechacambio	Timestamp		Campo de auditoria		
cali_comentario	Text		Guarda el comentario dado sobre un widget		

cali_respuesta	Text		Almacena la respuesta dada por un administrador a un comentario dado		
cali_fecha	Timestamp		Almacena la fecha en que se puntuó o se dio el comentario		
punt_id	Numeric	30	Identificador de la puntuación dada		Si
usua_id	Numeric	30	Identificador del usuario que hizo la puntuación y/o comentario		Si
vers_id	Numeric	30	Identificador de la versión del widget que se está calificando		Si

Tabla 58. Tabla dashboard.puntuacioncalificacion

Nombre de la tabla	puntuacioncalificacion				
Descripción	Guarda la relación de las puntuaciones dadas a una calificación				
Relaciones	Foráneo de la tabla dashboard.puntuacion Foráneo de la tabla dashboard.calificacion				
Campo	**Tipo**	**Tamaño**	**Descripción**	**Pk**	**Fk**
punt_id	Numeric	30	Almacena el identificador de la puntuación	Si	Si

cali_id	Numeric	30	Almacena el identificador de la calificación	Si	Si
puca_registradopor	Varchar	250	Campo de auditoria		
puca_fechacambio	Timestamp		Campo de auditoria		
puca_total	Numeric	10	Almacena el total de puntuaciones sobre la calificación, cuantas me gusta, cuantas poco útil, cuantas inadecuada, cuantas spam, …		

Tabla 59. Tabla dashboard.opinion

Nombre de la tabla	opinion				
Descripción	Guarda cada opinión dada a una calificación (me gusta, poco útil, inadecuada, spam)				
Relaciones	Foráneo de la tabla dashboard.puntuacion Foráneo de la tabla dashboard.usuario Foráneo de la tabla dashboard.calificacion				
Campo	**Tipo**	**Tamaño**	**Descripción**	**Pk**	**Fk**
opin_id	Numeric	30	Almacena el identificador de la calificación	Si	
opin_registradopor	Varchar	250	Campo de auditoria		
opin_fechacambio	Timestamp		Campo de auditoria		

opin_fecha	Timestamp		Feche en la cual fue hecho el comentario		
punt_id	Numeric	30	Identificador de la puntuación dada		Si
usua_id	Numeric	30	Identificador del usuario que dio la opinión		Si
cali_id	Numeric	30	Identificador de la calificación sobre la cual se está dando una opinión		Si

3.2.3 Modelo dinámico

El modelo dinámico se encarga de representar la forma en que el sistema responde a los eventos, describiendo este comportamiento por medio de diagramas de secuencia.

3.2.3.1 Diagramas de secuencia

En este apartado se encuentra los diagramas de secuencia para cada caso de uso, teniendo en cuenta el flujo principal y los alternativos. Es importante tener presente las líneas de vida utilizadas en estos, las cuales son actores, objeto de entidad, de control y de frontera. En la tabla 60 se presentan cada una de ellas

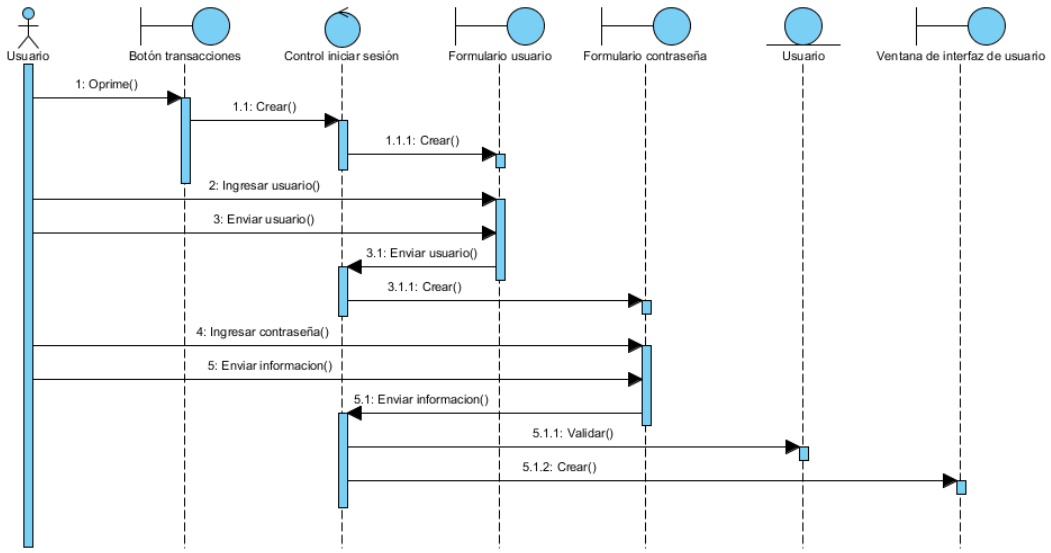
Tabla 60. Líneas de vida utilizadas en los diagramas de secuencia realizados

	Actor	Se encarga de definir un rol en el sistema
	Objeto de entidad	Este objeto hace referencia a la información persistente del sistema
	Objeto de control	Corresponde a las tareas que el sistema aborda
	Objeto de frontera	Se refiere a la interacción que se da entre el actor y el sistema

3.2.3.1.1 CU1. Iniciar sesión

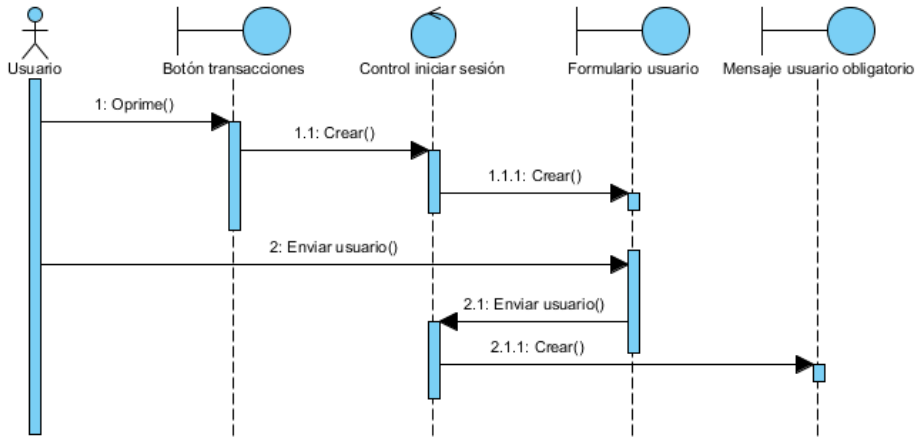
A continuación, en la figura 15 se presenta el diagrama de secuencia principal del caso de uso, iniciar sesión. En este, el usuario oprime el botón transacciones, el objeto de control crea el formulario usuario, se ingresa y se envía esta información. El objeto de control crea el formulario de la contraseña, se ingresa y se envía la contraseña. El sistema valida la información suministrada y crea la ventana de interfaz de usuario con las funcionalidades que le brinda la aplicación.

Figura 15. Diagrama de secuencia principal del caso de uso 1: iniciar sesión



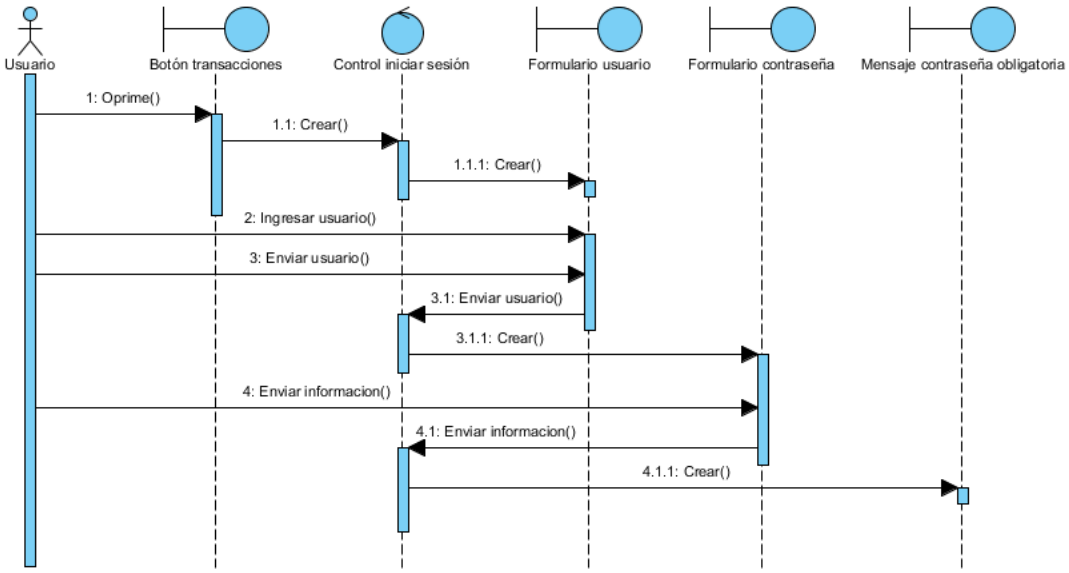
Este caso de uso cuenta con tres flujos alternativos. La figura 16 representa el primero. En este caso no es ingresado el usuario y el formulario es enviado, por ello, el istema muestra un mensaje indicando que este campo es obligatorio.

Figura 16. Diagrama de secuencia alternativo 1 del caso de uso 1: iniciar sesión



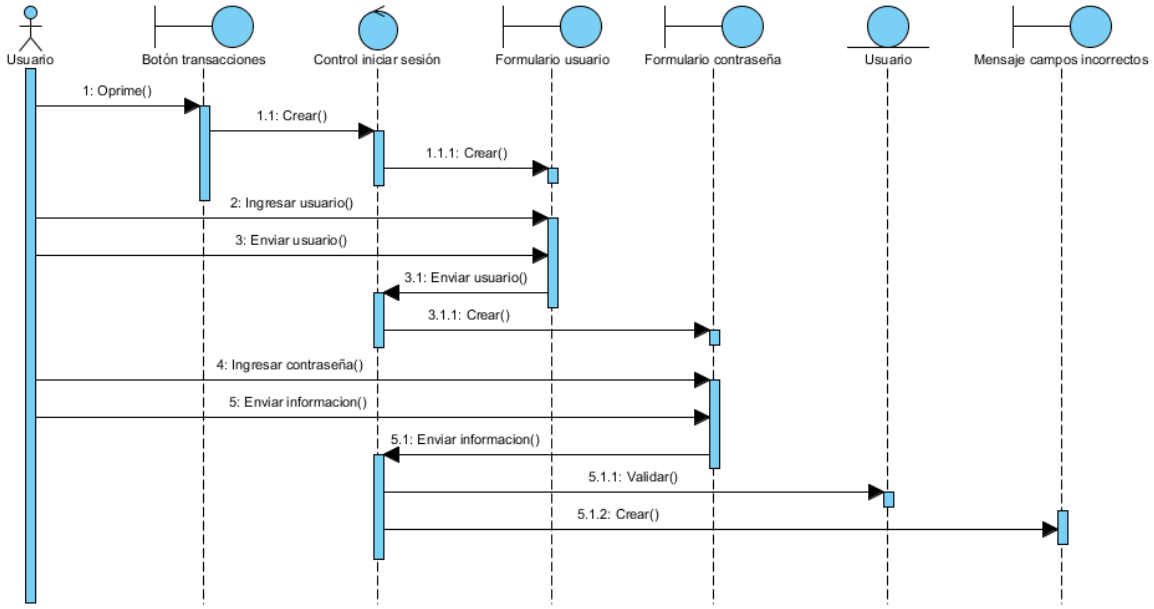
La figura 17 muestra el segundo diagrama de secuencia alternativo. Este se da ya que se es ingresado el usuario y enviado, pero no se ingresa la contraseña e igualmente el formulario es enviado. El sistema muestra un mensaje indicando que el campo contraseña es obligatorio.

Figura 17. Diagrama de secuencia alternativo 2 del caso de uso 1: iniciar sesión



El tercer flujo alternativo se da cuando el usuario ingresa la información correspondiente cada uno de los formularios, pero cuando el sistema intenta validarla, descubre que esta es incorrecta. En la figura 18 se muestra el diagrama de secuencia correspondiente para este.

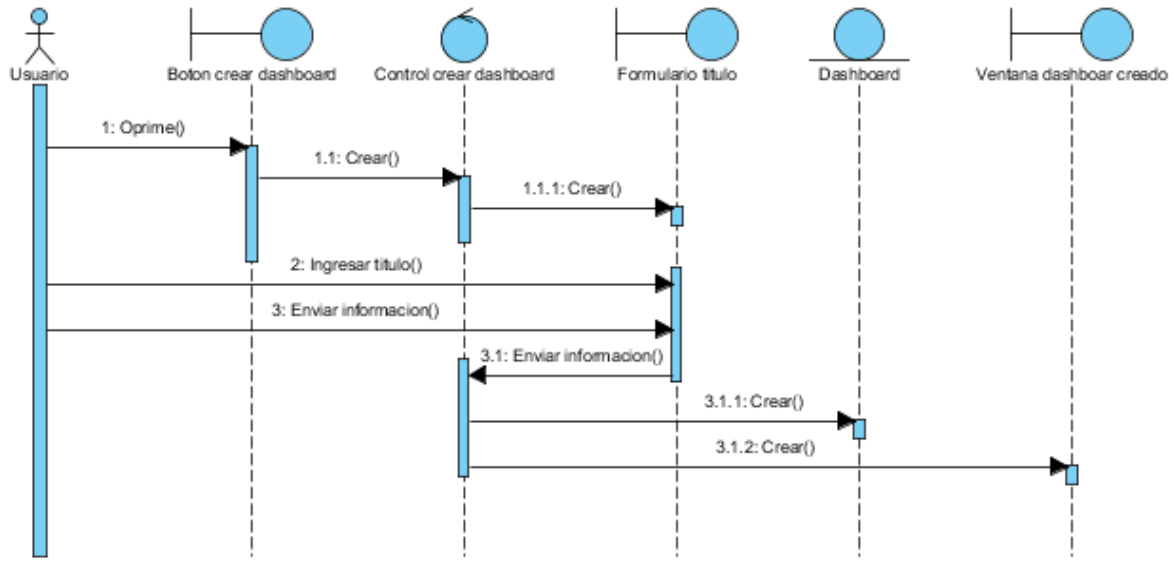
Figura 18. Diagrama de secuencia alternativo 3 del caso de uso 1: iniciar sesión



3.2.3.1.2 CU2. Crear *dashboard*

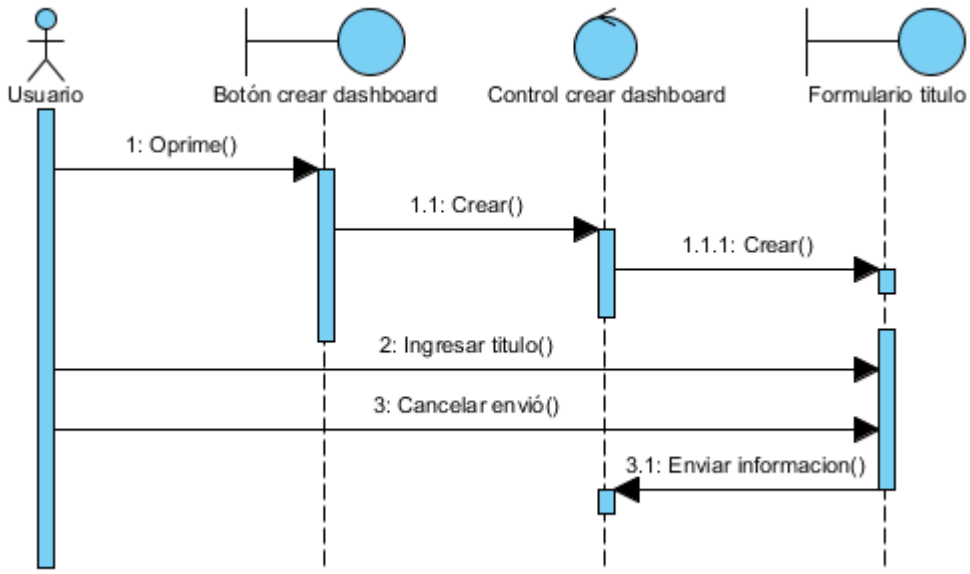
Los diagramas de secuencia que se muestran de la figura 19 a la 21 hacen parte del caso de uso crear *dashboard*. La figura 19 representa el flujo principal de este, en donde el usuario oprime el botón crear *dashboard*, se muestra el formulario que permite ingresar el título, el usuario lo ingresa y envía esta información. El sistema crea el dashboard en el objeto de entidad dashboard y muestra la ventana correspondiente a este.

Figura 19. Diagrama de secuencia principal del caso de uso 2: crear dashboard



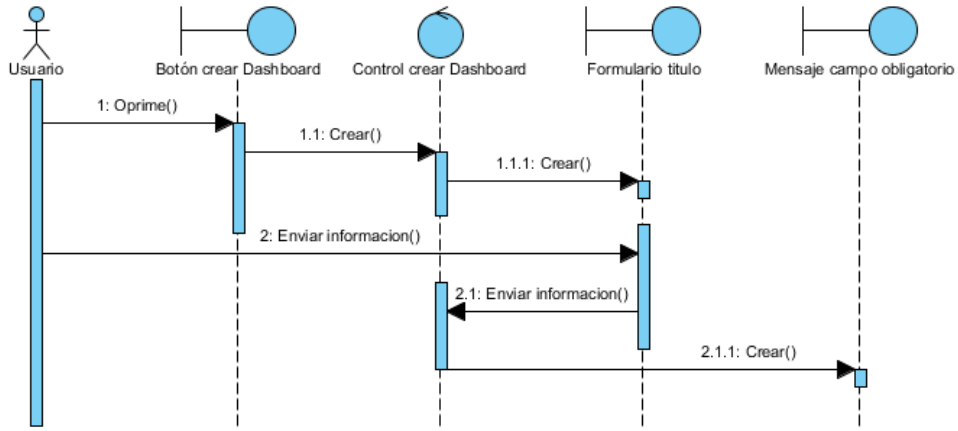
La figura 20 representa el primer flujo alternativo de este caso de uso. Este hace alusión a que el usuario ingresa el título que quiere para su nuevo *dashboard*, pero luego cancela esta operación.

Figura 20. Diagrama de secuencia alternativo 1 del caso de uso 2: crear dashboard



El segundo flujo alternativo está representado por la figura 21. En este, el usuario oprime el botón crear *dashboard*, apareciendo así el formulario que permite ingresar el título de este. El usuario realiza el envió, sin ingresar esta información, por lo tanto, el sistema muestra un mensaje indicando que este campo es obligatorio.

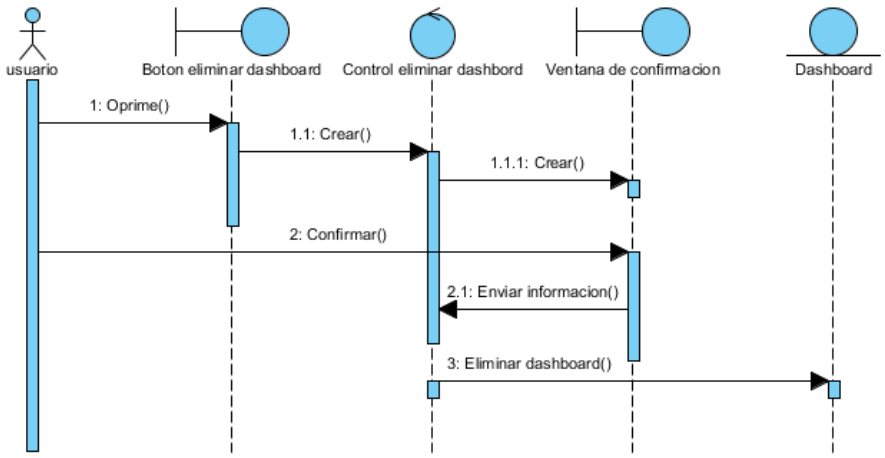
Figura 21. Diagrama de secuencia alternativo 2 del caso de uso 2: crear dashboard



3.2.3.1.3 CU3. Eliminar dashboard

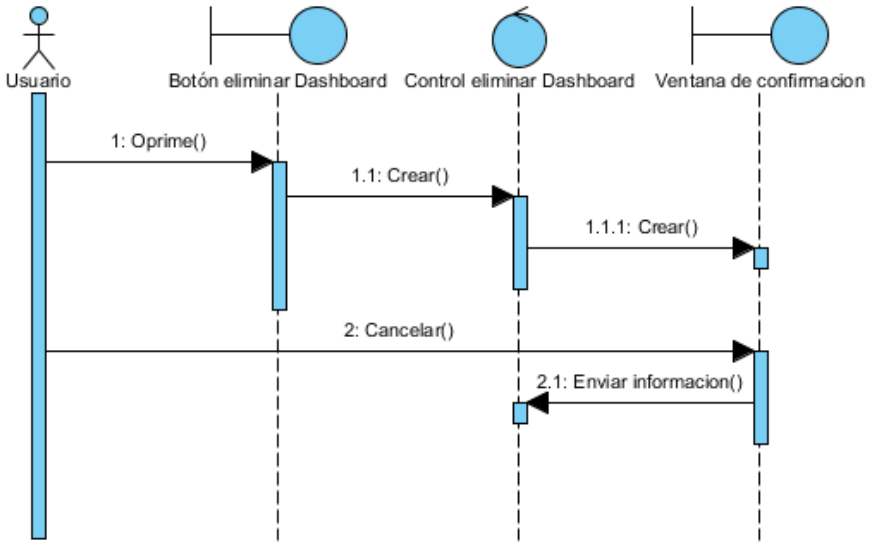
En la figura 22 se encuentra representado el flujo principal de este caso de uso. El usuario oprime el botón eliminar *dashboard y* el sistema muestra una ventana de confirmación, en donde se afirma esta acción. Por último, se elimina el dashboard del sistema.

Figura 22. Diagrama de secuencia principal del caso de uso 3: eliminar dashboard



La figura 23 muestra el flujo alternativo mediante un diagrama de secuencia. En este, el usuario cancela la acción de eliminar su dashboard.

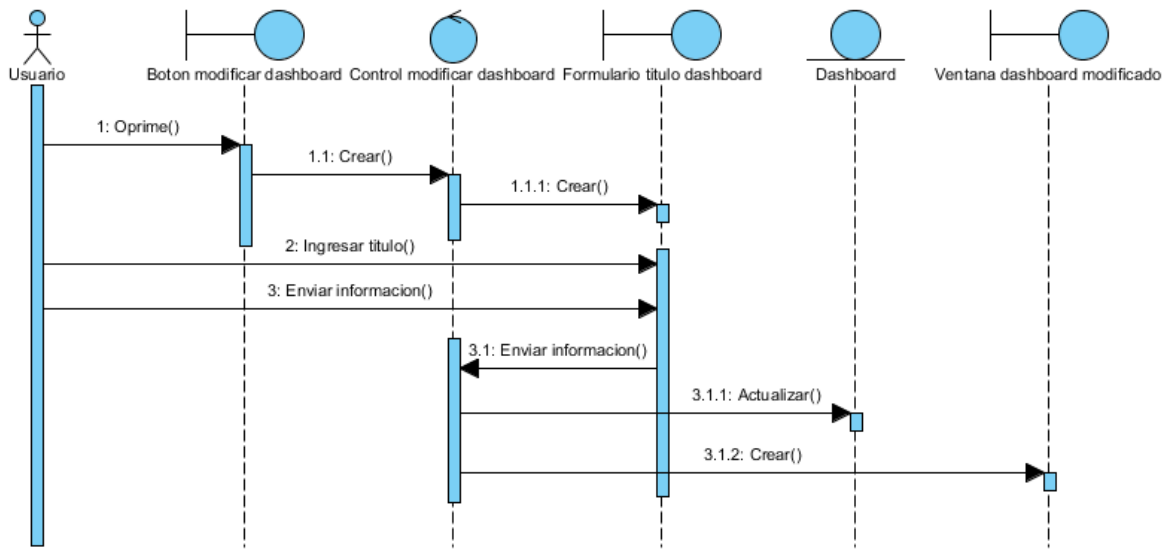
Figura 23. Diagrama de secuencia alternativo del caso de uso 3: eliminar dashboard



3.2.3.1.4 CU4. Modificar dashboard

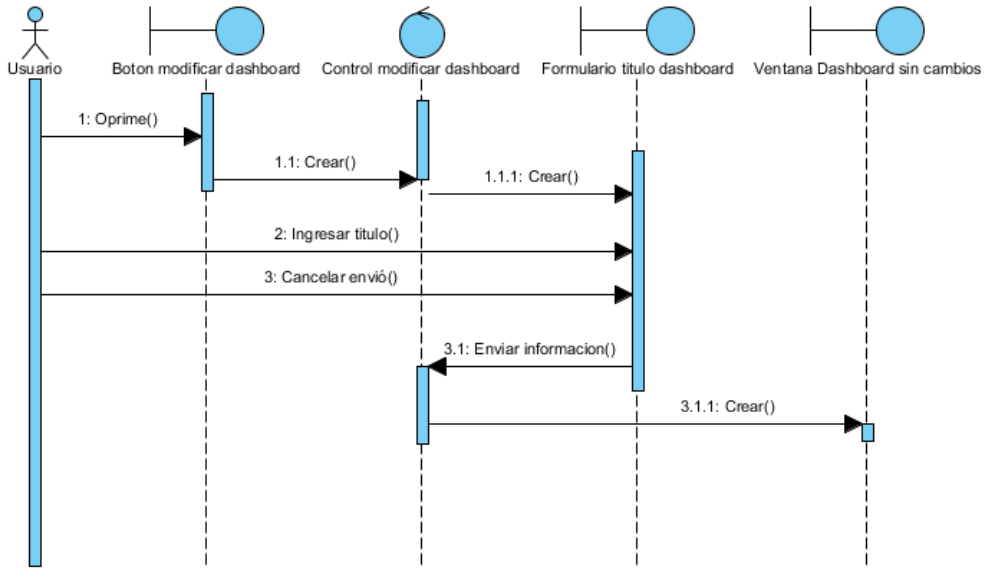
En la figura 24 se representa el diagrama de secuencia del flujo principal del caso de uso, modificar dashboard. El usuario oprime el botón modificar dashboard, se crea el formulario del título de este, el usuario lo modifica y envia la informacion. El sistema modifica el dashboard en el objeto de entidad y muestra la ventana actualizada.

Figura 24. Diagrama de secuencia principal del caso de uso 4: modificar dashboard



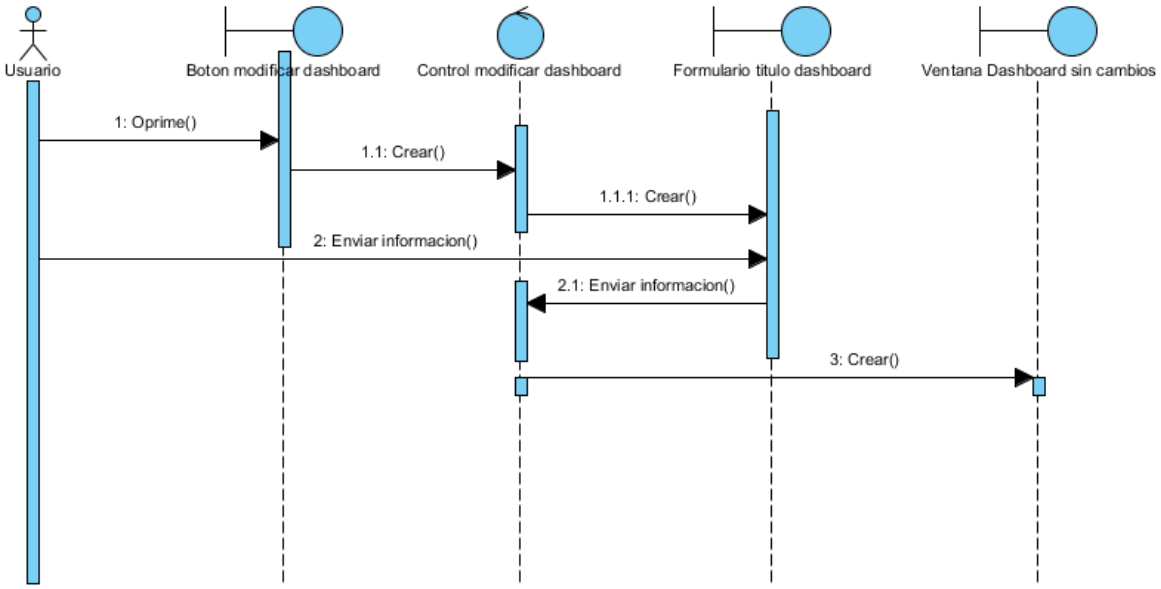
La figura 25 presenta el primer flujo alternativo de este caso de uso. En este, el usuario ingresa el nuevo título que tendrá su dashboard, pero no envía esta informacion sino cancela la solicitud. El sistema regresa al dashboard sin efectuar ningún cambio.

Figura 25. Diagrama de secuencia alternativo 1 del caso de uso 4: modificar dashboard



En este segundo caso alternativo, el usuario oprime modificar dashboard, no ingresa la información a actualizar y envía así el formulario. Por lo tanto, el sistema no efectúa ningún cambio. El diagrama de secuencia que representa esto se encuentra en la figura 26

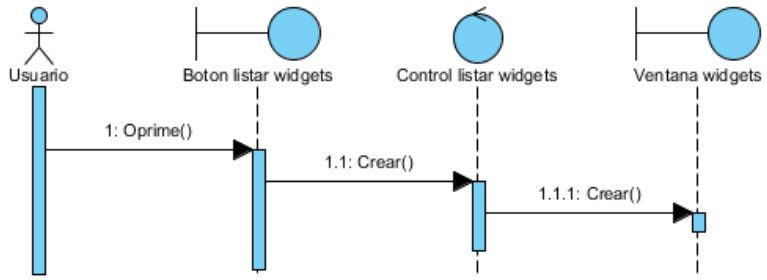
Figura 26. Diagrama de secuencia alternativo 2 del caso de uso 4: modificar dashboard



3.2.3.1.5 CU5. Listar widgets

A continuación, se presenta mediante la figura 27, el diagrama de secuencia principal del caso de uso, listar widgets. En este, el usuario oprime listar widgets y aparece una ventana con los que se encuentran disponibles.

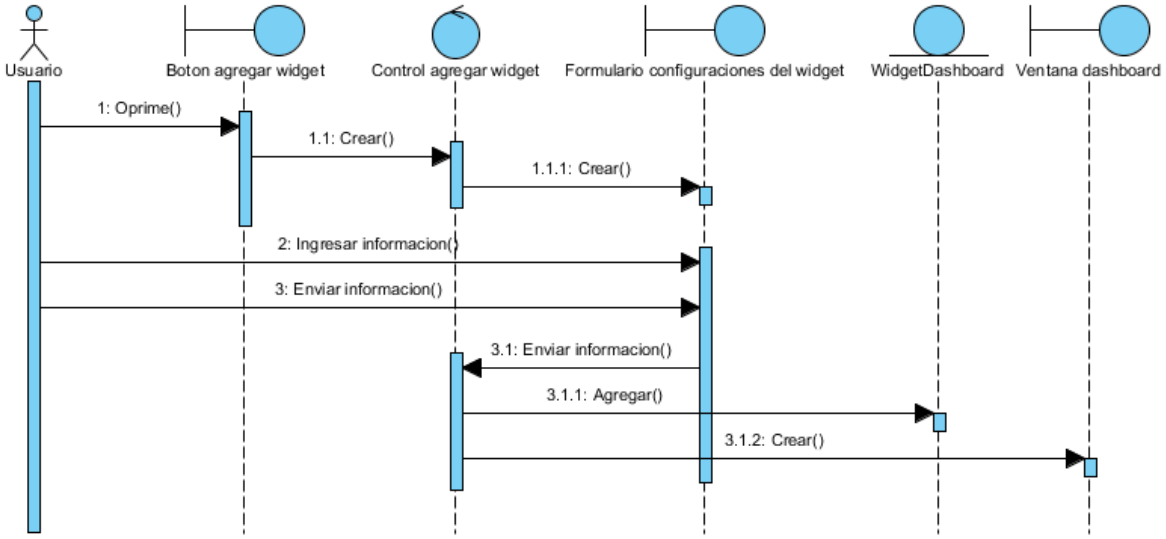
Figura 27. Diagrama de secuencia principal del caso de uso 5: listar widgets



3.2.3.1.6 CU6. Agregar widget

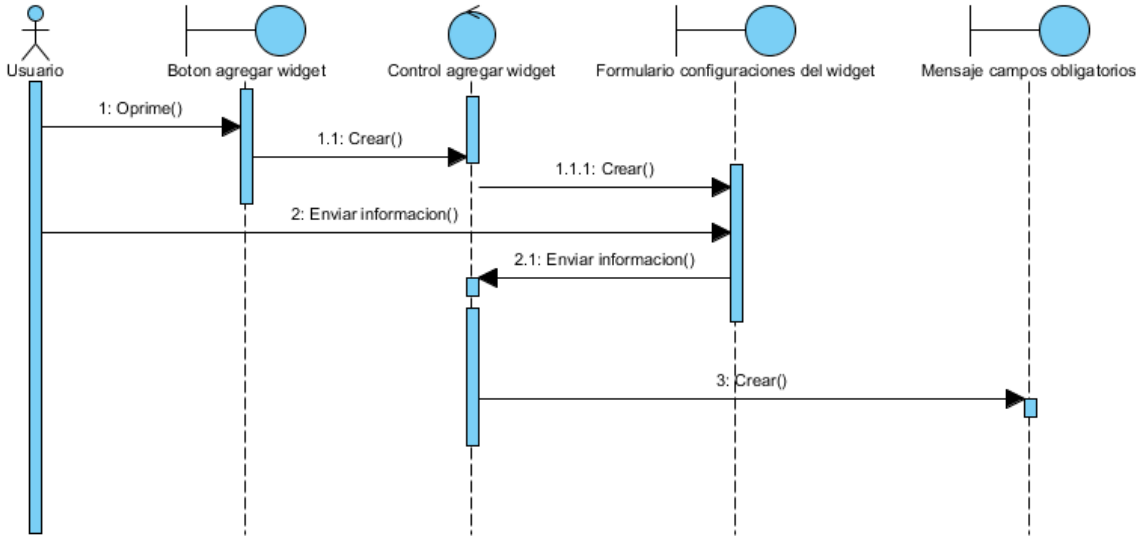
El flujo principal de este caso de uso, consiste en que el usuario oprime el botón agregar widget, a continuación, se crea el formulario que permite ingresar las configuraciones de este, el usuario ingresa los campos del mismo y envía esta información. El sistema agrega este nuevo widget en el objeto de entidad y lo muestra en pantalla. Esto se evidencia en la figura 28.

Figura 28. Diagrama de secuencia principal del caso de uso 6: agregar widget



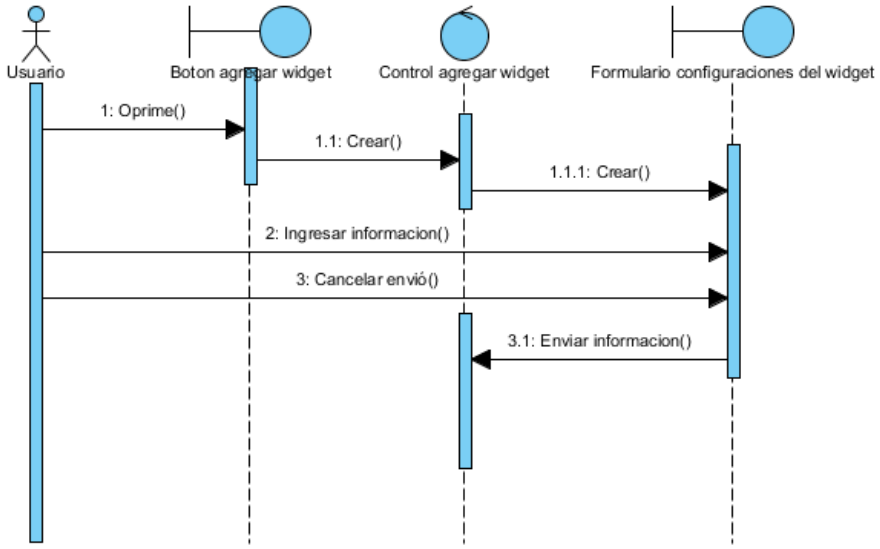
La figura 29 muestra el primer diagrama de secuencia alternativo de este caso de uso. En este, el usuario oprime el botón agregar widget y se crea el formulario de las configuraciones. Este es enviado sin suministrar ningún tipo de información, por lo que el sistema muestra un mensaje indicando que los campos son obligatorios.

Figura 29. Diagrama de secuencia alternativo 1 del caso de uso 6: agregar widget



Otro caso es cuando, al oprimir la opción agregar widget por parte del usuario este ingresa la información correspondiente, pero seguido de esto cancela el envió de dicha información. Este segundo flujo alternativo es representado mediante la figura 30.

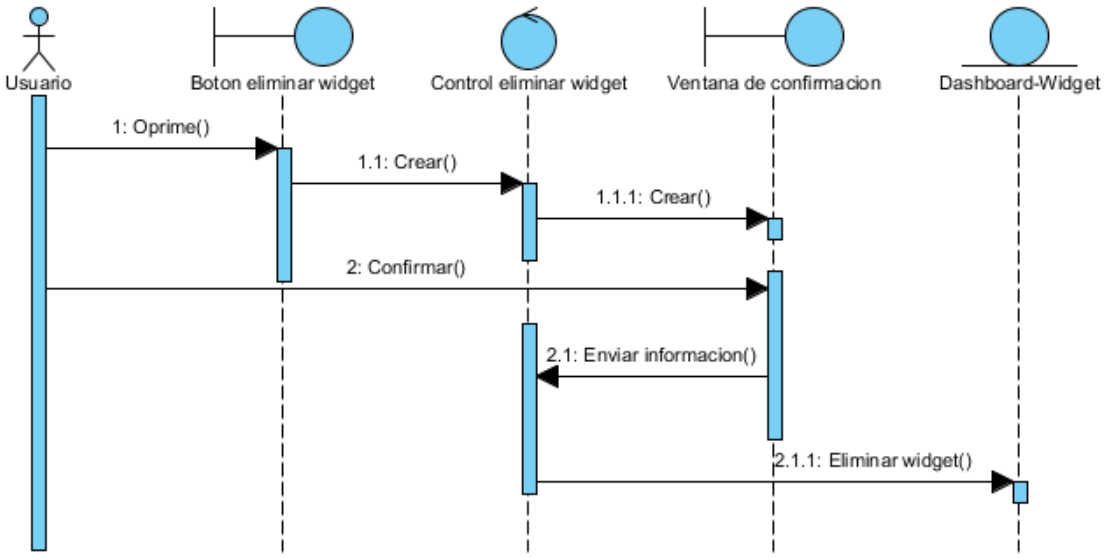
Figura 30. Diagrama de secuencia alternativo 2 del caso de uso 6: agregar widget



3.2.3.1.7 CU7. Eliminar widget

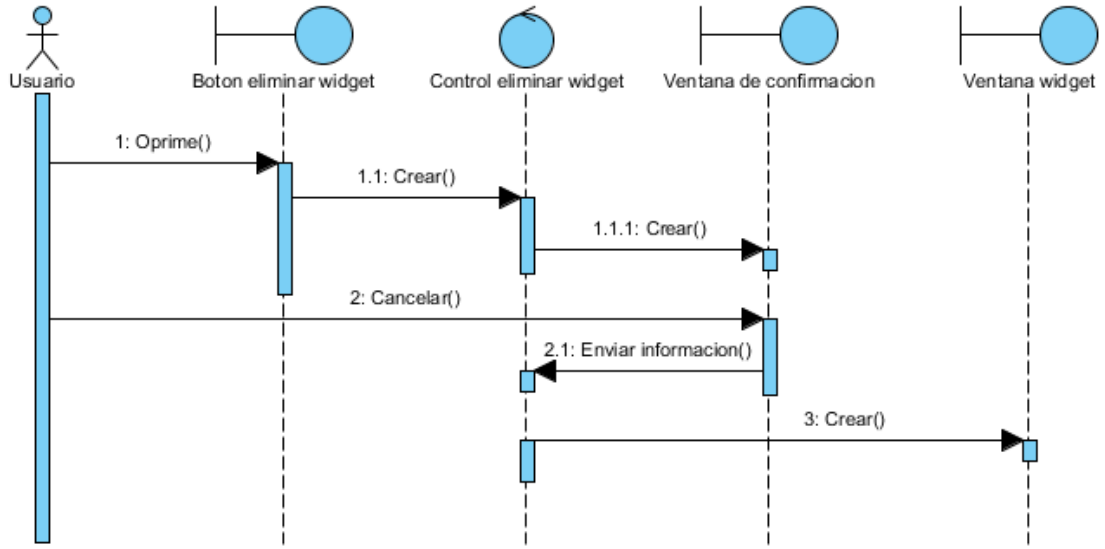
La figura 31 muestra el flujo principal de este caso de uso, mediante un diagrama de secuencia. Este inicia cuando el usuario oprime el botón eliminar widget. Al realizar esta acción aparece una ventana de confirmación, el usuario afirma que desea eliminar dicho widget y el sistema lo elimina del objeto de entidad.

Figura 31. Diagrama de secuencia principal del caso de uso 7: eliminar widget



El diagrama de secuencia alternativo de este caso de uso se presenta en la figura 32. El usuario cancela la acción de eliminar el widget de su *dashboard*, por lo tanto, el sistema vuelve a la ventana del mismo.

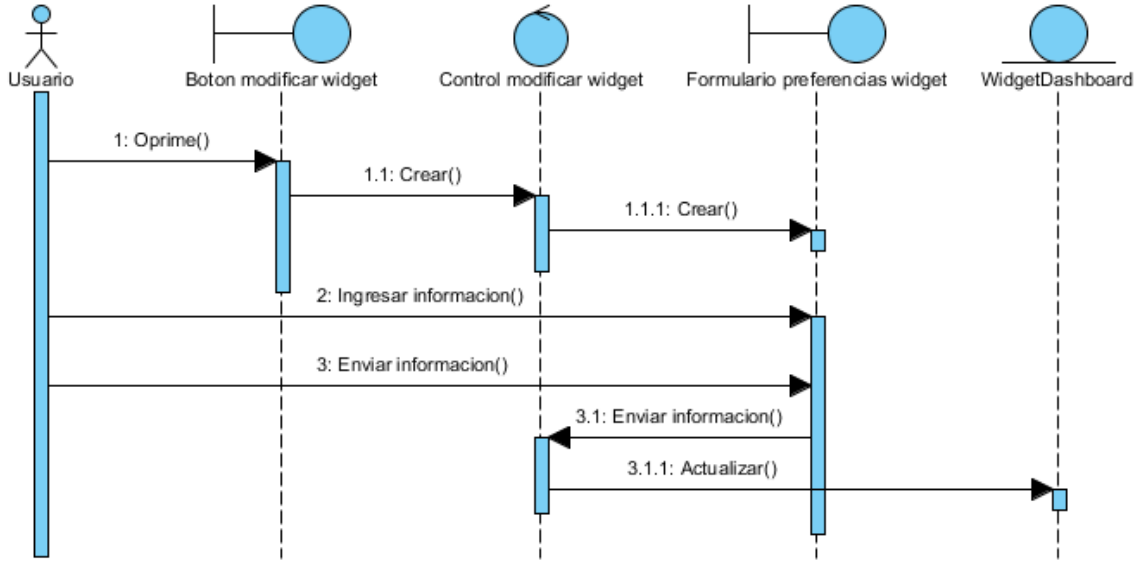
Figura 32. Diagrama de secuencia alternativo del caso de uso 7: eliminar widget



3.2.3.1.8 CU8. Modificar widget

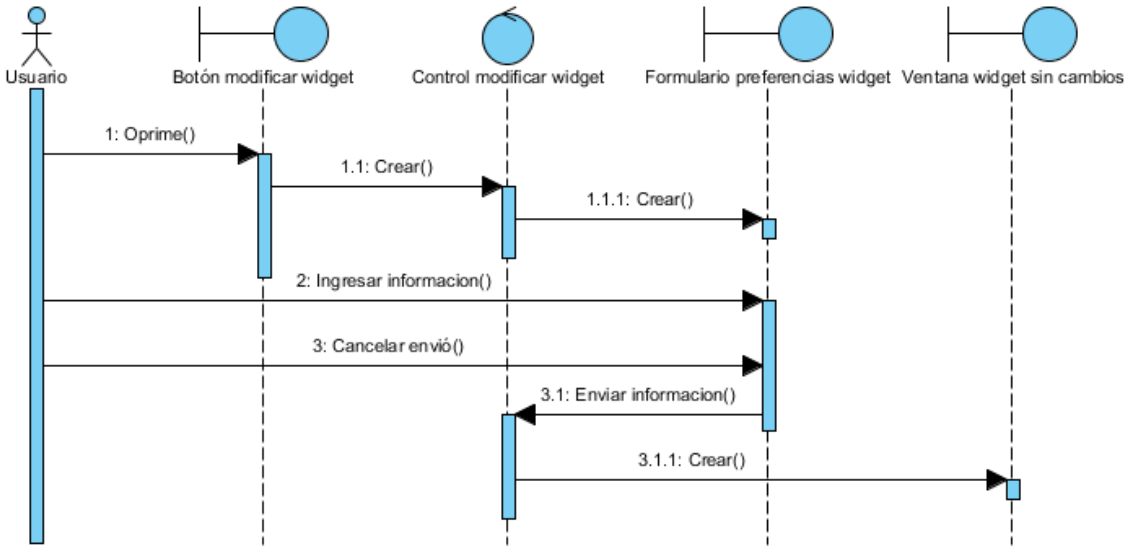
Los diagramas de secuencia que se muestran de la figura 33 a la 35, constituyen el caso de uso modificar widget. La figura 33 muestra el flujo principal de este. El usuario oprime el botón modificar widget, lo cual permite que el objeto de control cree el formulario de configuración del widget. Se ingresa la información a actualizar y se envía. Por último, el sistema la modifica.

Figura 33. Diagrama de secuencia principal del caso de uso 8: modificar widget



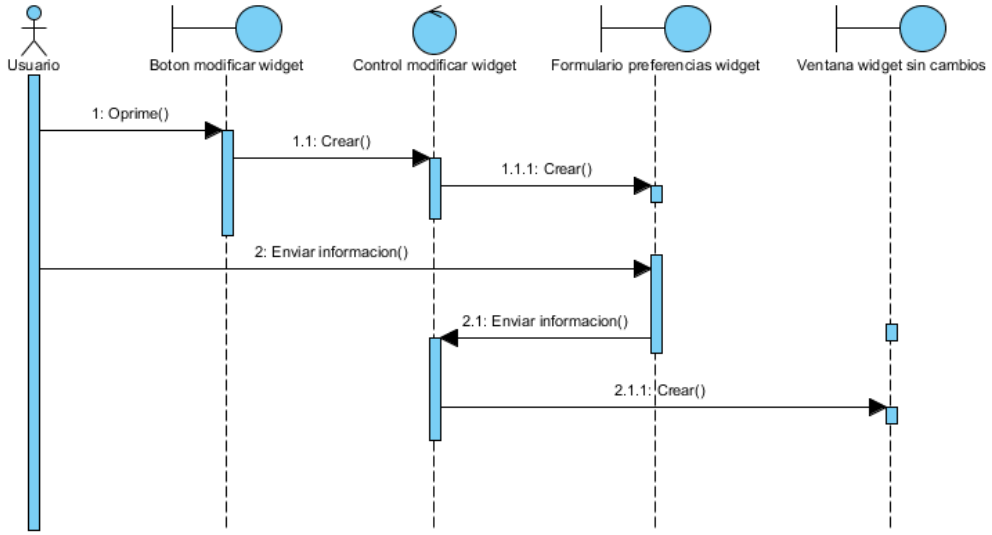
La figura 34 muestra el diagrama de secuencia correspondiente al primer flujo alternativo de este caso de uso. El usuario oprime el botón modificar widget, al realizar esto se crea el formulario de las configuraciones del mismo. Se ingresa la nueva información, pero se cancela el envió. El sistema muestra el widget sin efectuar ningún cambio.

Figura 34. Diagrama de secuencia alternativo 1 del caso de uso 8: modificar widget



El segundo diagrama de secuencia alternativo se evidencia en la figura 35. En este, el usuario no ingresa la nueva informacion requerida para realizar la actualización del widget, envía este formulario y por lo tanto el sistema muestra el widget sin efectuar ningún cambio.

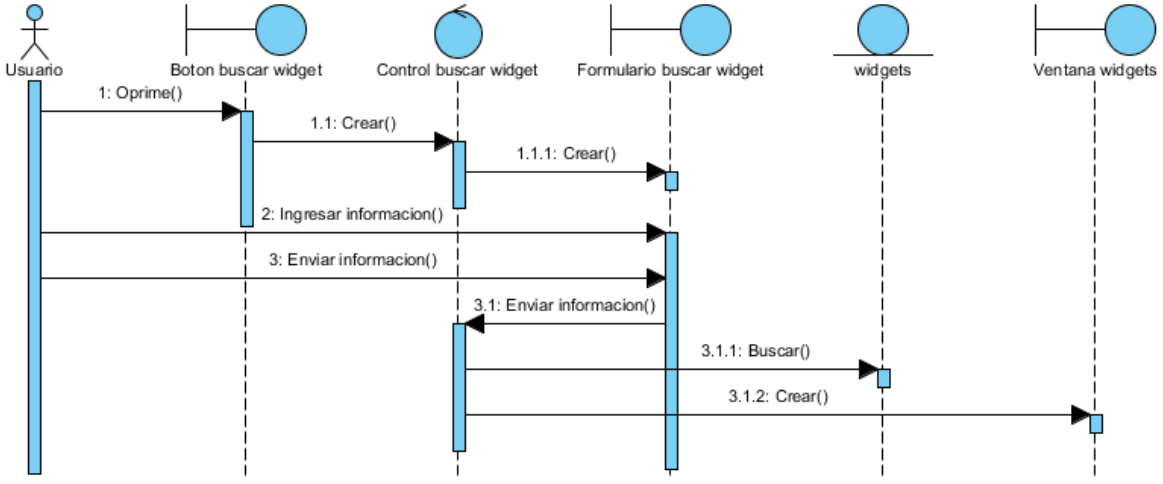
Figura 35. Diagrama de secuencia alternativo 2 del caso de uso 8: modificar widget



3.2.3.1.9 CU9. Buscar widget

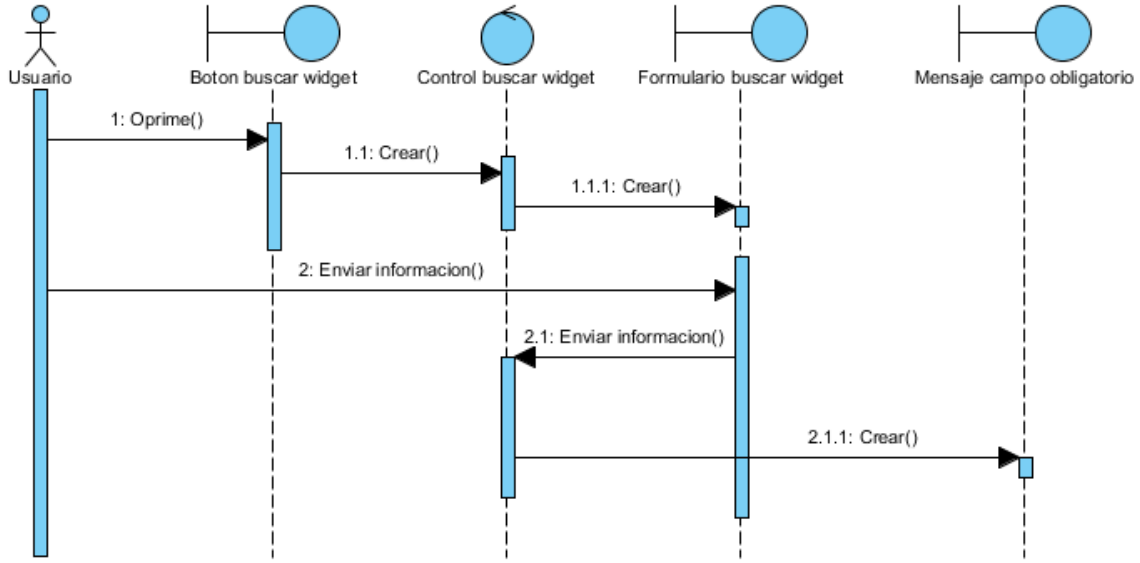
La figura 36 representa el flujo principal de este caso de uso, mediante un diagrama de secuencia. El usuario oprime el botón buscar widget, aparece el formulario que permite ingresar el nombre a buscar. Al realizar el envio de esta informacion, el sistema realiza esta búsqueda y muestra una ventana con los resultados de la misma.

Figura 36. Diagrama de secuencia principal del caso de uso 9: buscar widget



El caso alternativo para este caso de uso, se presenta cuando el usuario no ingresa la informacion del formulario y realiza así el envío de la misma. Por tanto, el sistema muestra un mensaje indicando que el campo es obligatorio.

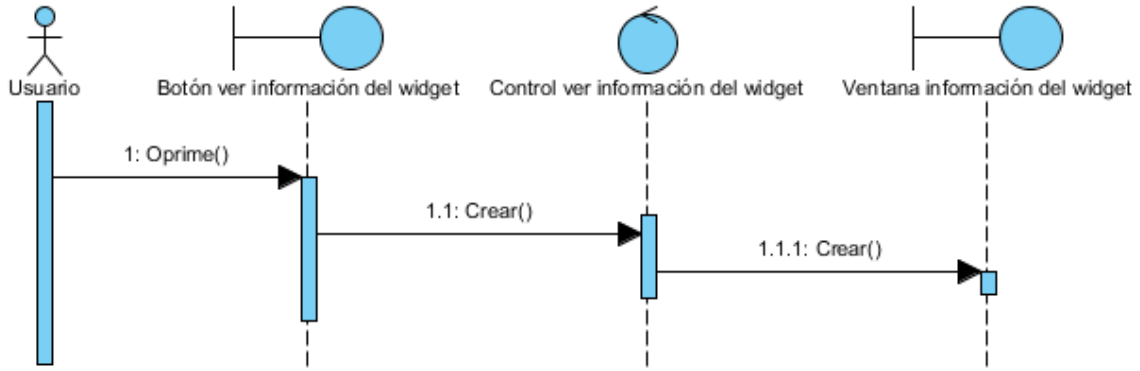
Figura 37. Diagrama de secuencia alternativo del caso de uso 9: buscar widget



3.2.3.1.10 CU10. Visualizar widget

Este caso de uso permite al usuario visualizar la informacion de un widget en específico, al oprimir el botón ver informacion del widget. En la figura 38 se presenta el diagrama de secuencia correspondiente

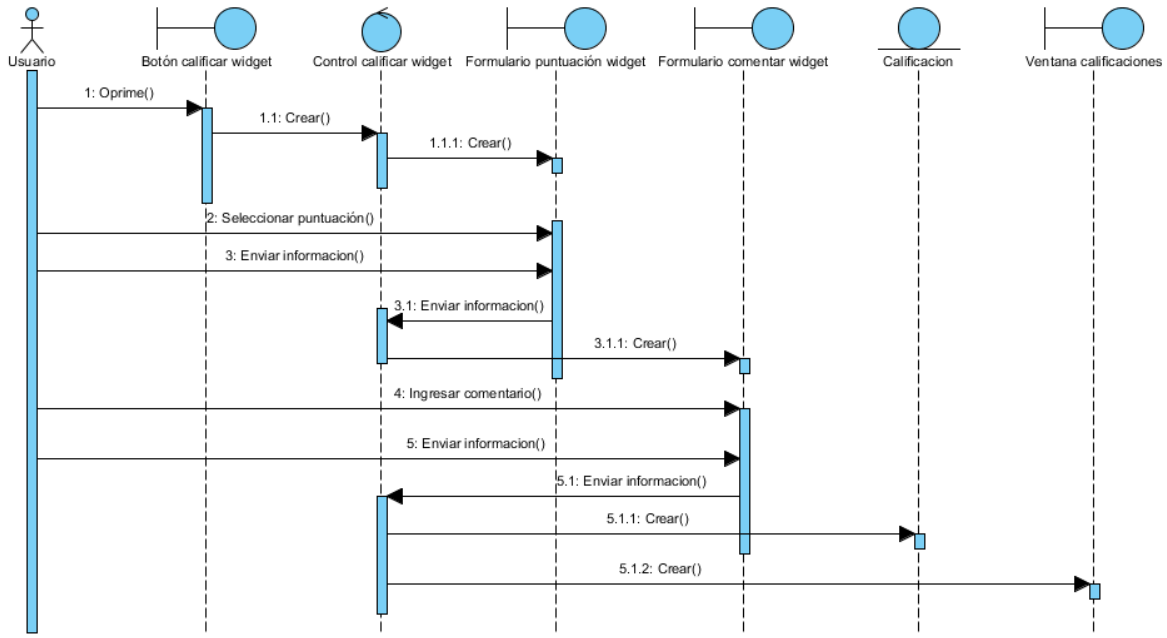
Figura 38. Diagrama de secuencia principal del caso de uso 10: visualizar widget



3.2.3.1.11 CU11. Agregar calificación

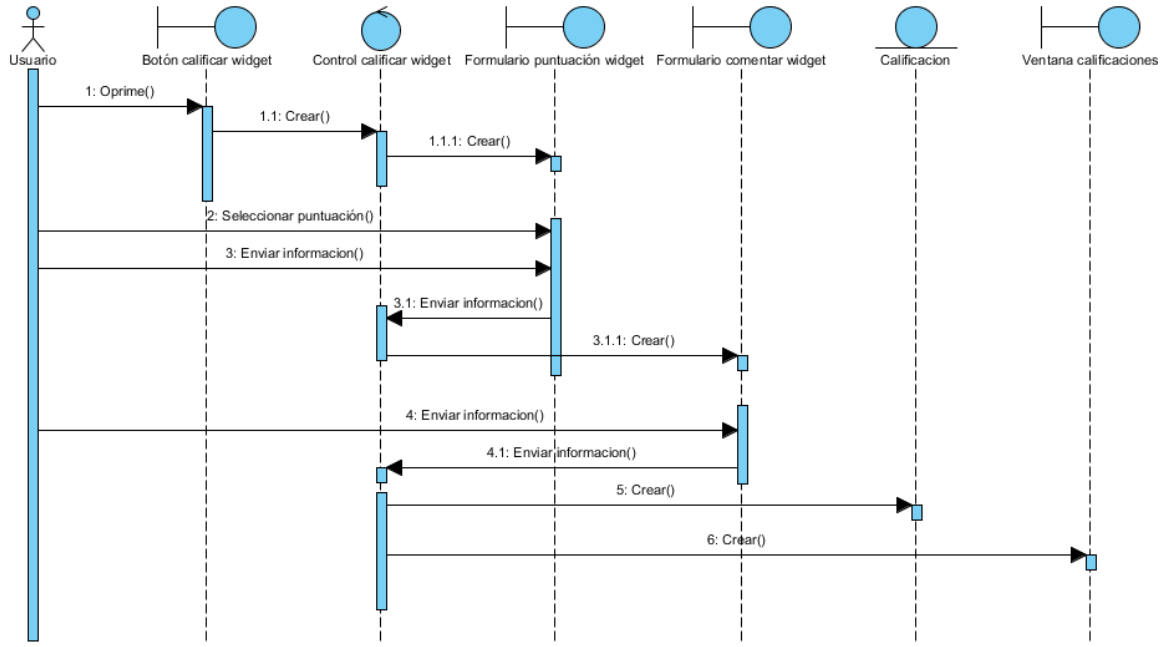
En las figuras 39 a la 43 se presentan los diagramas de secuencia correspondientes a este caso de uso. El flujo principal se refleja en la figura 39. En este, el usuario oprime el botón calificar widget, permitiendo así que se cree el formulario que permite ingresar la puntuación, el usuario ingresa esta informacion y la envía. En seguida, se permite ingresar el comentario correspondiente para la calificación, este es ingresado y enviado. El sistema inserta la informacion suministrada y muestra una ventana con esta.

Figura 39. Diagrama de secuencia principal del caso de uso 11: agregar calificación



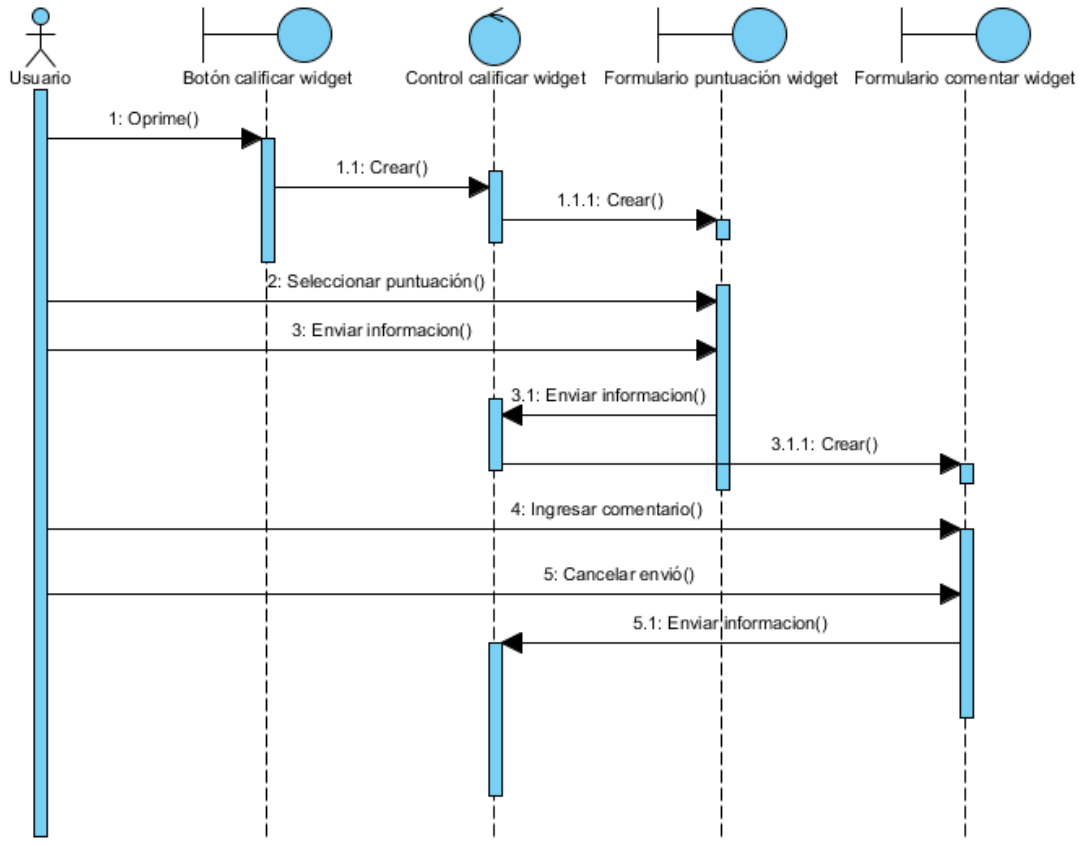
El siguiente diagrama de secuencia alternativo representa cuando el usuario agrega una nueva calificación, sin ingresar el comentario correspondiente a esta, es decir, ingresando únicamente la puntuación. En este caso el sistema almacena la información suministrada y luego muestra una ventana con esta. Esto se visualiza en la figura 40.

Figura 40. Diagrama de secuencia alternativo 1 del caso de uso 11: agregar calificación



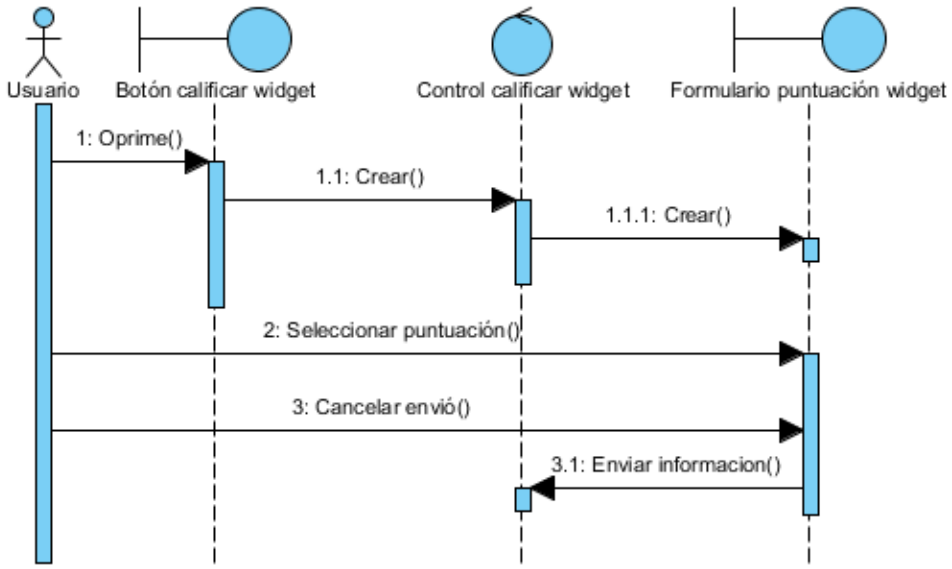
La figura 41 se encarga de mostrar el segundo flujo alternativo de este caso de uso. En el cual, el usuario después de ingresar la puntuación y el comentario, cancela el envio de este. En este caso el sistema no realiza ninguna acción.

Figura 41. Diagrama de secuencia alternativo 2 del caso de uso 11: agregar calificación



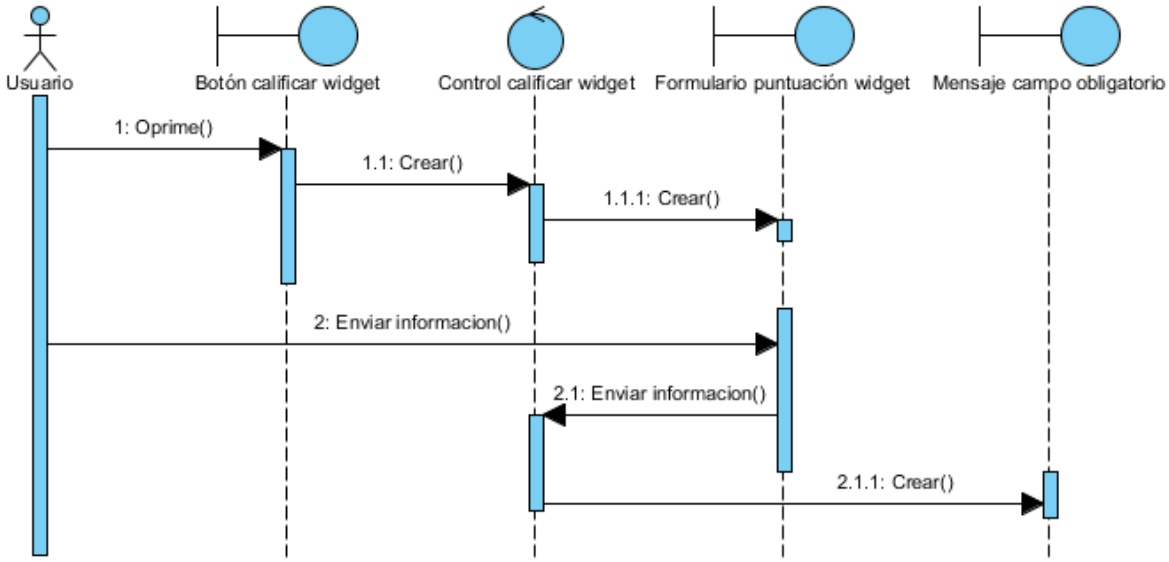
La figura 42 hace referencia al tercer diagrama de secuencia alternativo de este caso de uso. El usuario después de seleccionar la puntuación, cancela el envio de esta informacion, generando que el sistema no almacene ningún cambio.

Figura 42. Diagrama de secuencia alternativo 3 del caso de uso 11: agregar calificación



El último flujo alternativo se ve representado en la figura 43. En este el usuario intenta enviar el formulario de la puntuación, sin haber ingresado una previamente, esto ocasiona que el sistema muestre un mensaje indicando que este campo es obligatorio.

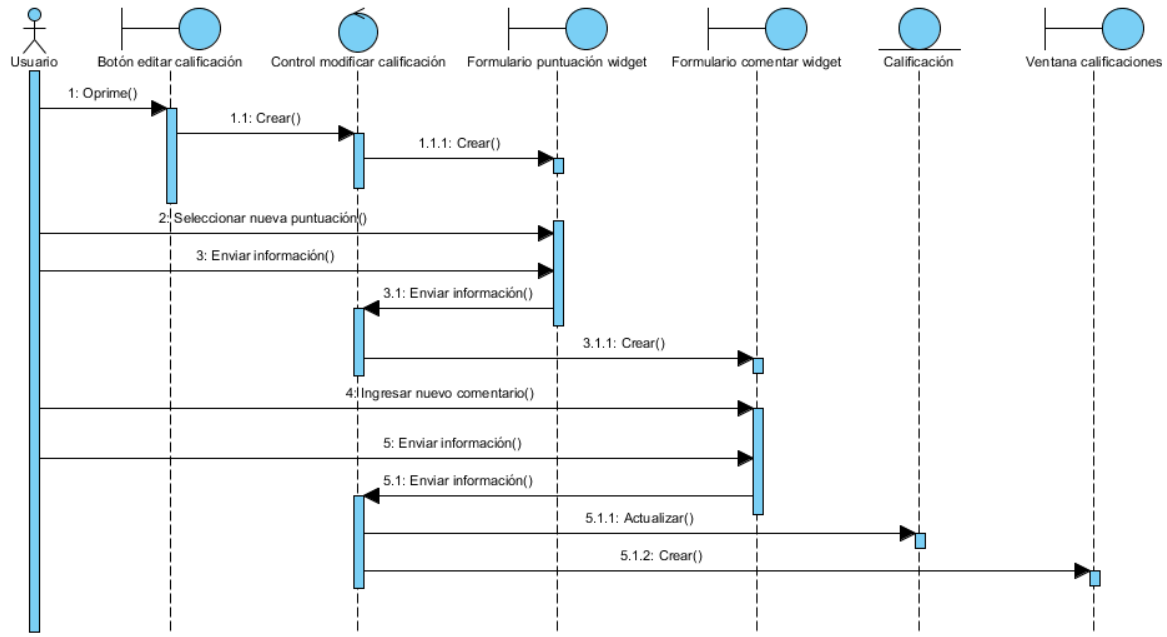
Figura 43. Diagrama de secuencia alternativo 4 del caso de uso 11: agregar calificación



3.2.3.1.12 CU12. Modificar calificación

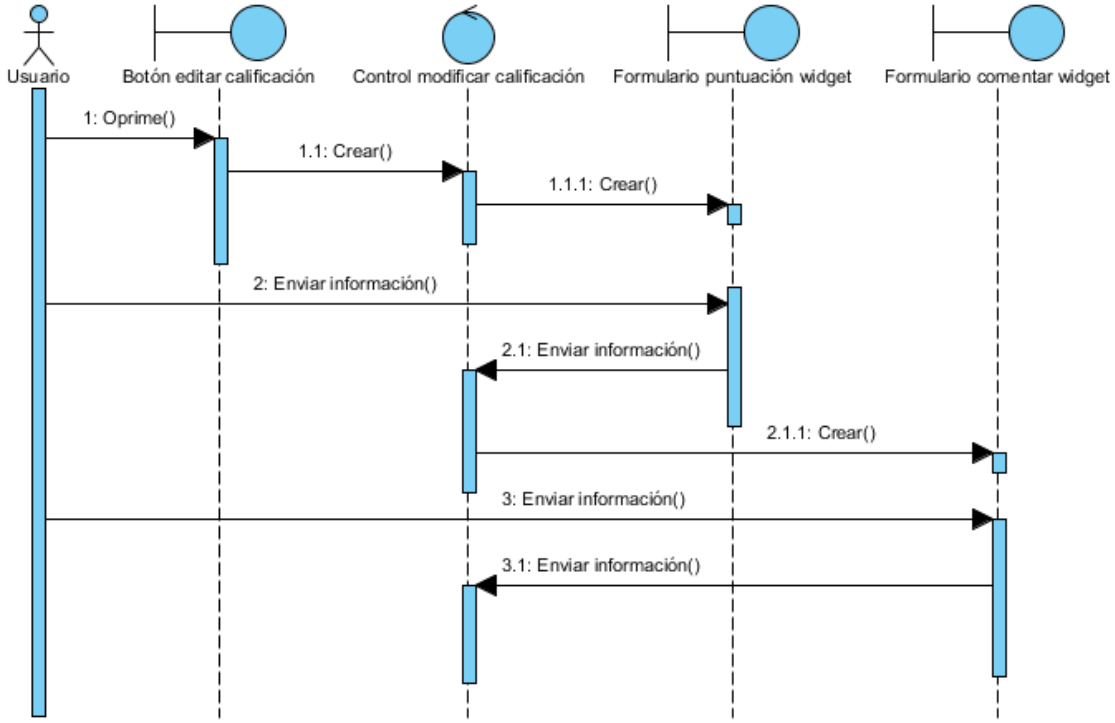
La figura 44 se encarga de mostrar el diagrama de secuencia correspondiente al flujo principal de este caso de uso. El usuario oprime editar calificación, el objeto de control crea el formulario de la puntuación del widget, se ingresa la nueva puntuación y se envía esta. El objeto de control crea el formulario del comentario, el usuario ingresa el nuevo comentario y lo envía. El sistema actualiza esta información y muestra una ventana con la calificación modificada.

Figura 44. Diagrama de secuencia principal del caso de uso 12: modificar calificación



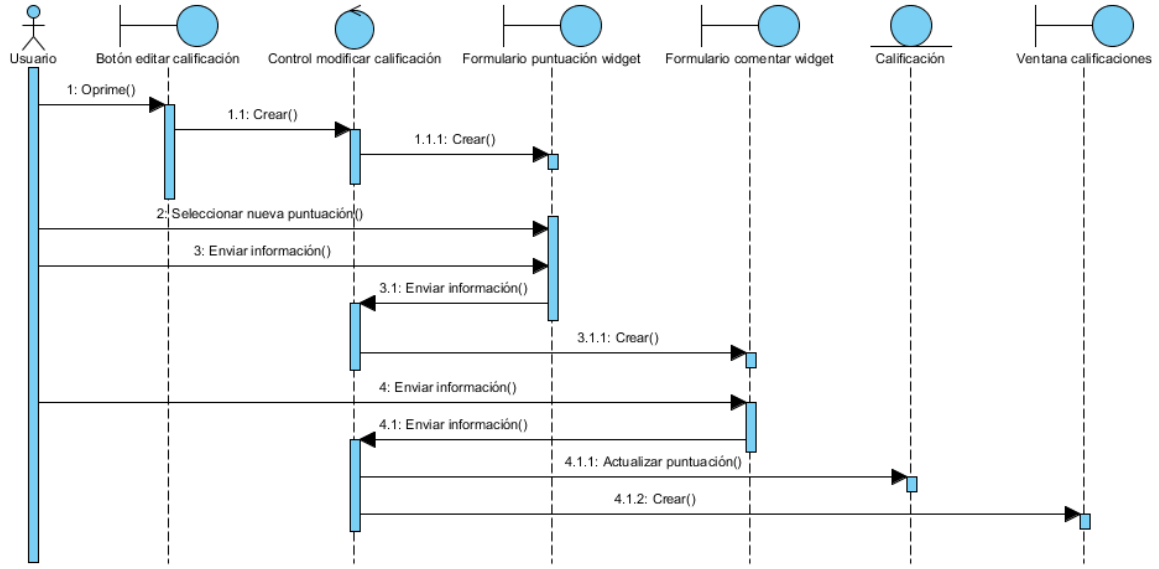
La figura 45 muestra el primer diagrama de secuencia alternativo de modificar calificación. En este se observa que el usuario oprime el botón editar calificación, creándose así el formulario de la puntuación del widget, el usuario hace envió de este sin ingresar una nueva puntuación. Lo mismo sucede con el formulario del comentario del widget. Por tanto, el sistema no efectúa ningún cambio.

Figura 45. Diagrama de secuencia alternativo 1 del caso de uso 12: modificar calificación



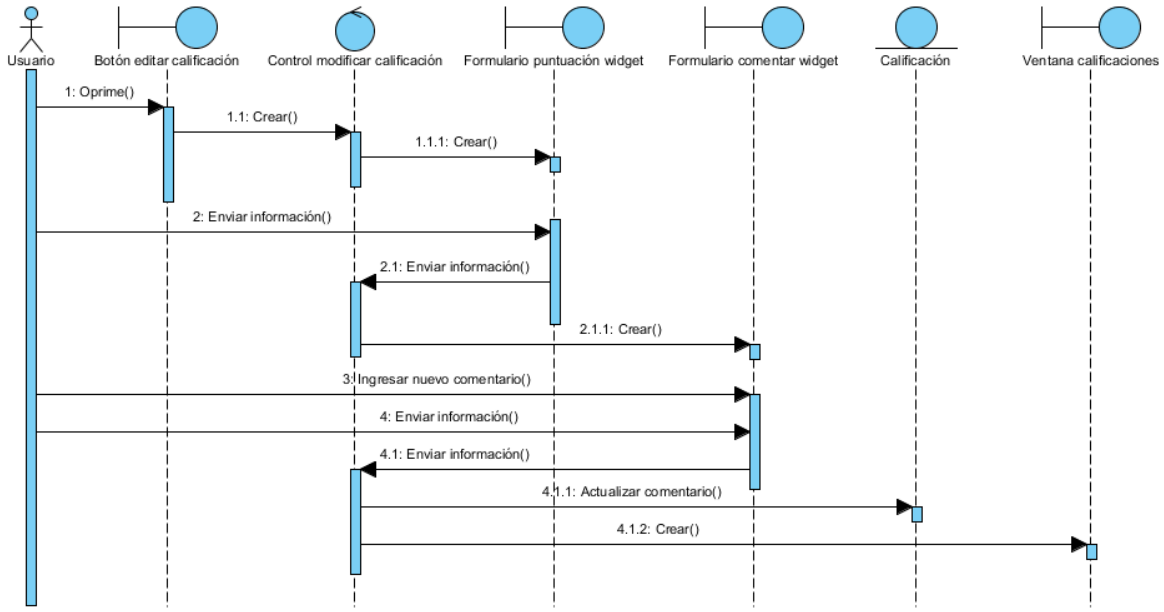
En la figura 46 se ve reflejado el segundo diagrama de secuencia alternativo. Al crearse el formulario de puntuación, el usuario ingresa el nuevo valor que desea. En este caso no modifica el comentario de dicha calificación. El sistema actualiza la información suministrada y la muestra en pantalla.

Figura 46. Diagrama de secuencia alternativo 2 del caso de uso 12: modificar calificación



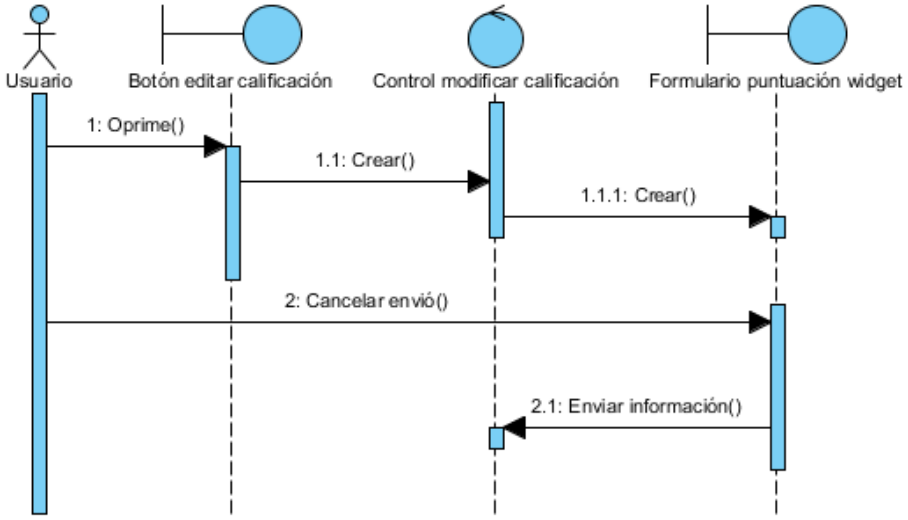
Otro caso alternativo se da cuando el usuario no modifica la puntuación de la calificación, pero si el comentario. En este caso el sistema actualiza esta información y muestra una ventana con la misma. La figura 47 representa el diagrama de secuencia correspondiente

Figura 47. Diagrama de secuencia alternativo 3 del caso de uso 12: modificar calificación



El ultimo flujo alternativo se da cuando el usuario inmediatamente después de oprimir el botón de editar calificación, cancela el envió de la información. Este está representado mediante un diagrama de secuencia en la figura 48.

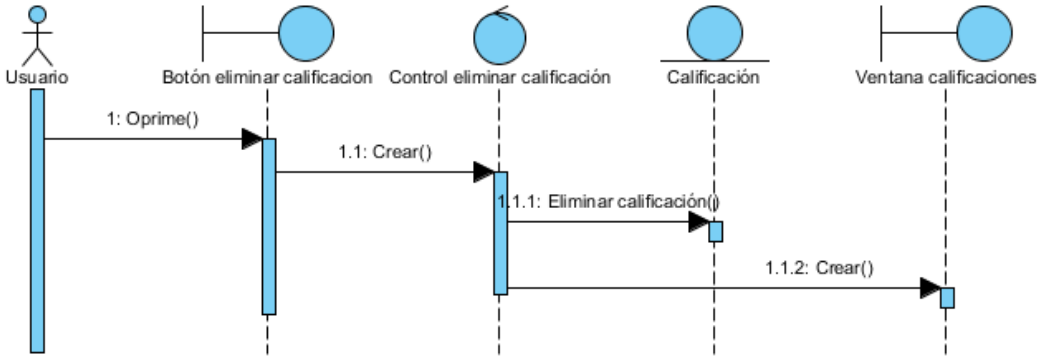
Figura 48. Diagrama de secuencia alternativo 4 del caso de uso 12: modificar calificación



3.2.3.1.13 CU13. Eliminar calificación

Cuando el usuario oprime el botón eliminar calificación, el sistema la elimina del objeto de entidad calificación y vuelve a la ventana anterior. El diagrama de secuencia correspondiente al flujo principal de este caso de uso se encuentra en la figura 49

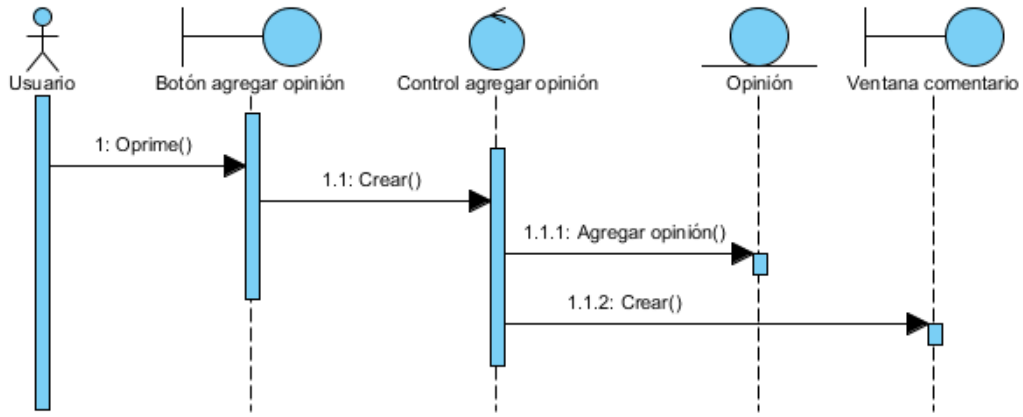
Figura 49. Diagrama de secuencia principal del caso de uso 13: eliminar calificación



3.2.3.1.14 CU14. Agregar opinión

La figura 50 representa el flujo principal de este caso de uso, esto lo realiza mediante un diagrama de secuencia. El usuario oprime agregar opinión y el objeto de control la agrega en el objeto de entidad. El sistema muestra el comentario con la opinión agregada.

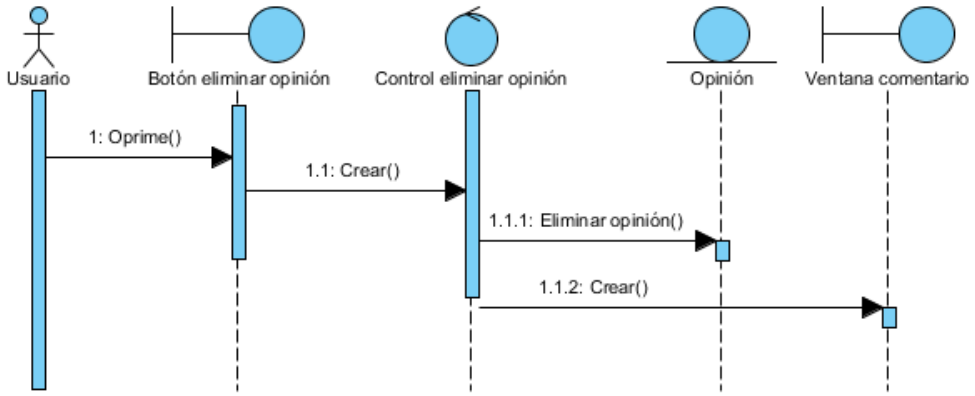
Figura 50. Diagrama de secuencia principal del caso de uso 14: Agregar opinión



3.2.3.1.15 CU15. Eliminar opinión

Este caso de uso permite que el usuario pueda eliminar la opinión realizada previamente sobre un comentario. Para esto oprime eliminar opinión, el sistema la elimina y muestra el comentario sin esta. La figura 51 representa mediante el diagrama de secuencia lo anteriormente dicho.

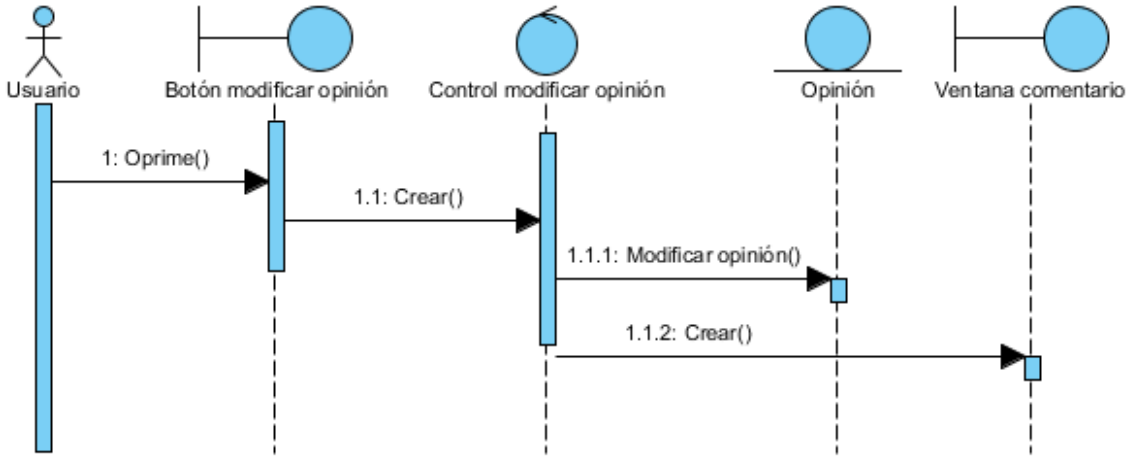
Figura 51. Diagrama de secuencia principal del caso de uso 15: Eliminar opinión



3.2.3.1.16 CU16. Modificar opinión

A continuación, en la figura 52 se presenta el flujo principal de este caso de uso. El usuario oprime modificar opinión, el sistema la modifica y muestra el comentario con esta informacion actualizada.

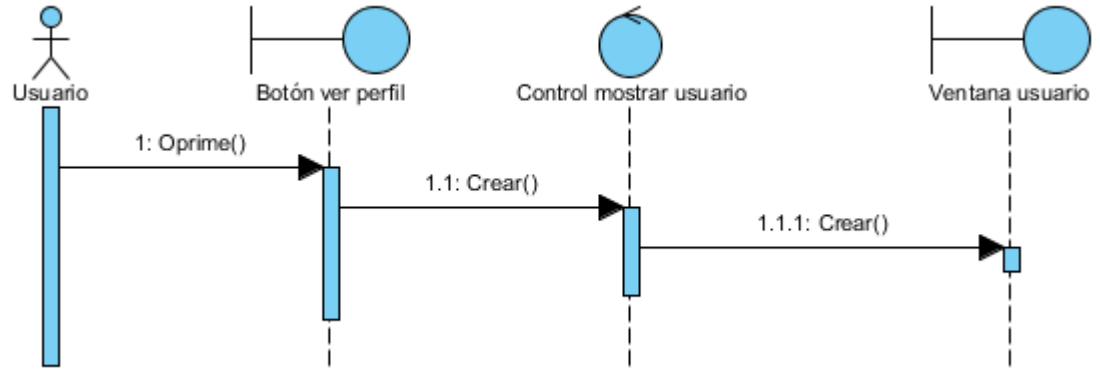
Figura 52. Diagrama de secuencia principal del caso de uso 16: modificar opinión



3.2.3.1.17 CU17. Visualizar usuario

El usuario oprime el botón ver perfil, seguido de esto, el sistema muestra una ventana con la informacion del mismo. En la figura 53 se presenta el diagrama de secuencia correspondiente.

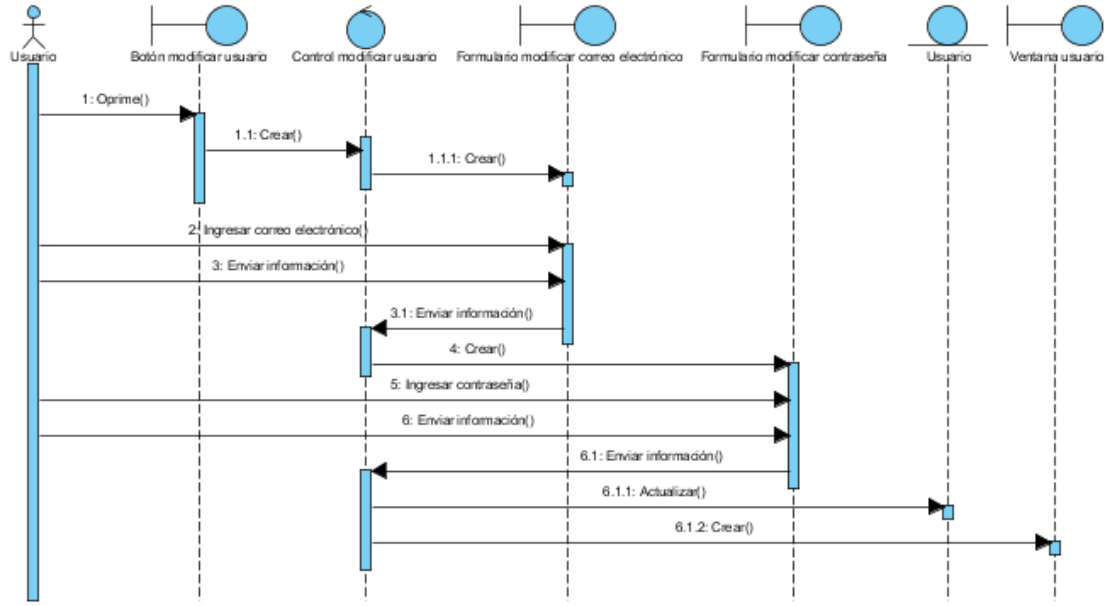
Figura 53. Diagrama de secuencia principal del caso de uso 17: visualizar usuario



3.2.3.1.18 CU18. Modificar usuario

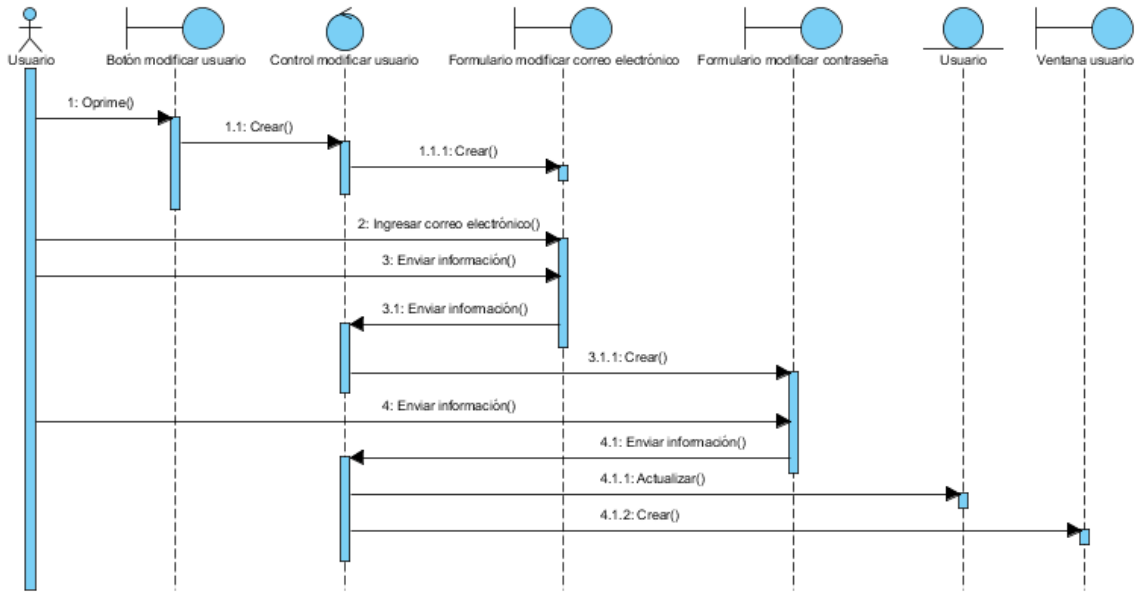
El flujo principal de este caso de uso está representado en la figura 54, en el cual, el usuario oprime el botón modificar usuario y se crea el formulario del correo electrónico. Él ingresa el nuevo correo electrónico y lo envía. A continuación, se crea el formulario de la contraseña, igualmente es ingresada y enviada. El sistema actualiza esta información y muestra la ventana del usuario con la nueva información.

Figura 54. Diagrama de secuencia principal del caso de uso 18: modificar usuario



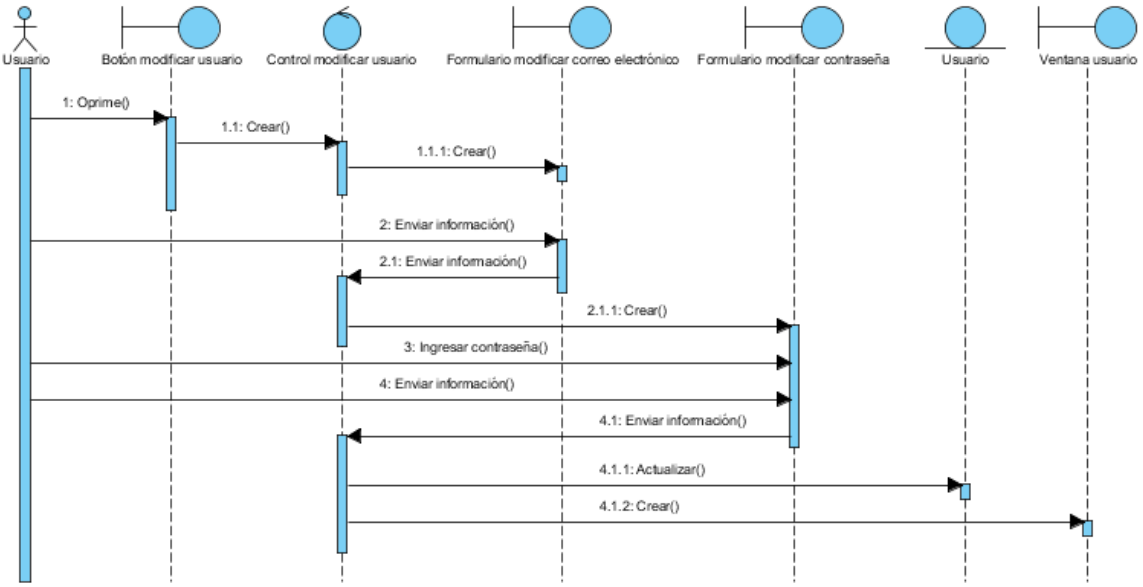
El primer flujo alternativo de este caso de uso, se presenta cuando el usuario quiere modificar únicamente su correo electrónico, el sistema se encarga de actualizarlo y luego muestra la ventana correspondiente al usuario. La figura 55 muestra lo anteriormente dicho.

Figura 55. Diagrama de secuencia alternativo 1 del caso de uso 18: modificar usuario



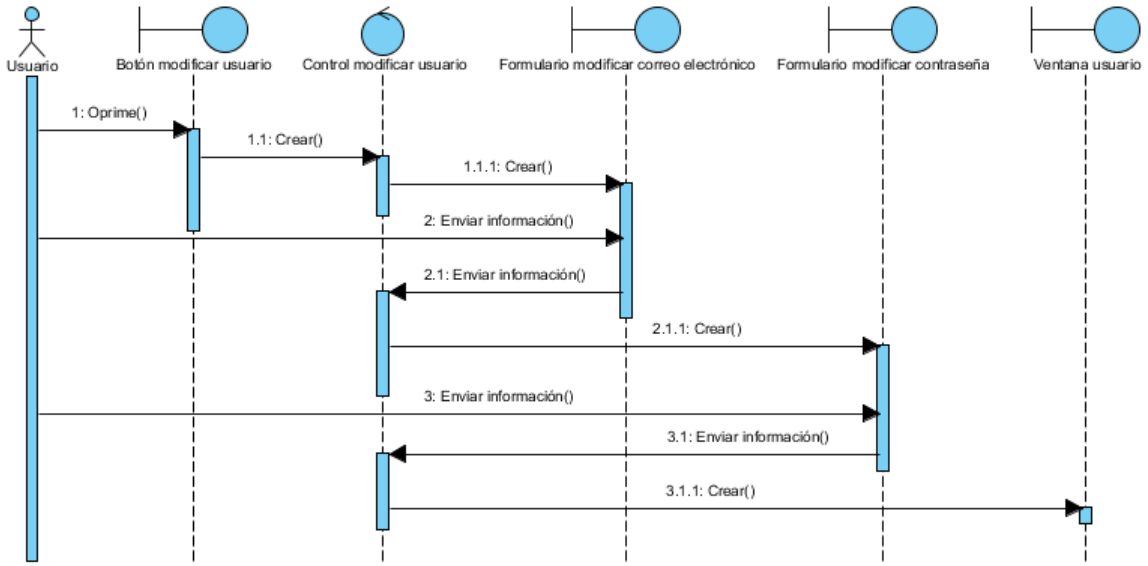
La figura 56 permite observar el segundo flujo alternativo mediante el diagrama de secuencia. En este caso el usuario solo desea modificar la contraseña, por tanto, el sistema la actualiza y muestra la ventana de usuario.

Figura 56. Diagrama de secuencia alternativo 2 del caso de uso 18: modificar usuario



Este último diagrama de secuencia alternativo del caso de uso modificar usuario, hace referencia a cuando el usuario oprime el botón de modificar usuario, pero al crearse los formularios no ingresa informacion nueva. Por ello el sistema no realiza ninguna actualización, en cambio muestra la ventana del usuario sin ningún cambio realizado. Esto se evidencia en la figura 57.

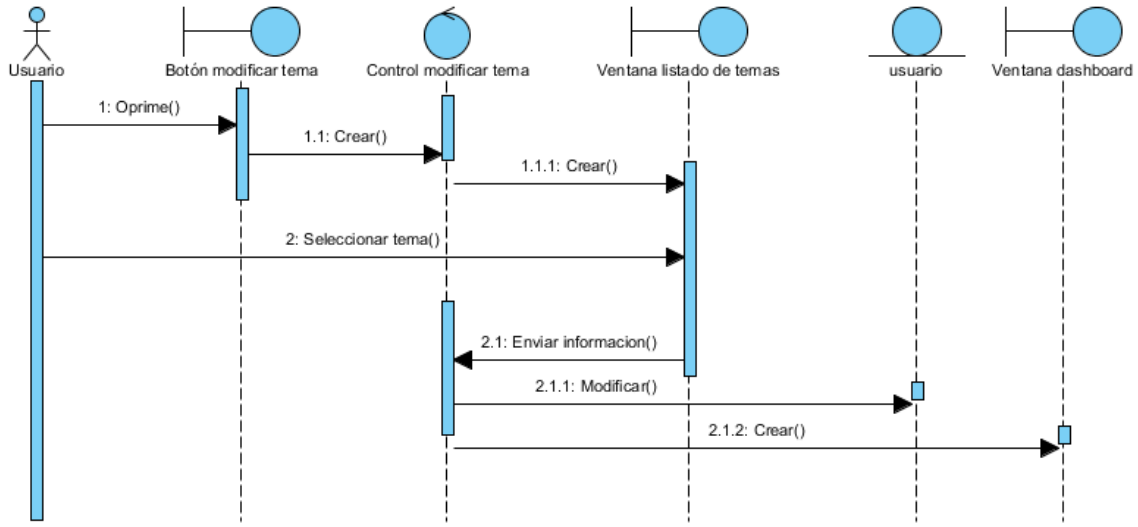
Figura 57. Diagrama de secuencia alternativo 3 del caso de uso 18: modificar usuario



3.2.3.1.19 CU19. Modificar tema

A continuación, en la figura 58, se tiene el flujo principal de este caso de uso, representado mediante un diagrama de secuencia. El usuario oprime el botón modificar tema, se crea una ventana con el listado de temas disponibles y es seleccionado el tema deseado por el usuario. El sistema almacena esta informacion y la modifica en la interfaz.

Figura 58. Diagrama de secuencia principal del caso de uso 19: modificar tema



3.2.4 Interfaz de usuario

Como se dijo en el capítulo 2 correspondiente al marco de referencia, la interfaz de usuario tiene que ver con la parte visible de un software, siendo esta la que permite, que el usuario y el sistema se comuniquen. La maquetación de las pantallas principales de la aplicación se puede observar en el siguiente capítulo, numeral 4.2.

4 DISEÑO DEL SISTEMA

En este capítulo se describe el diseño de alto nivel, que corresponde a la arquitectura del sistema

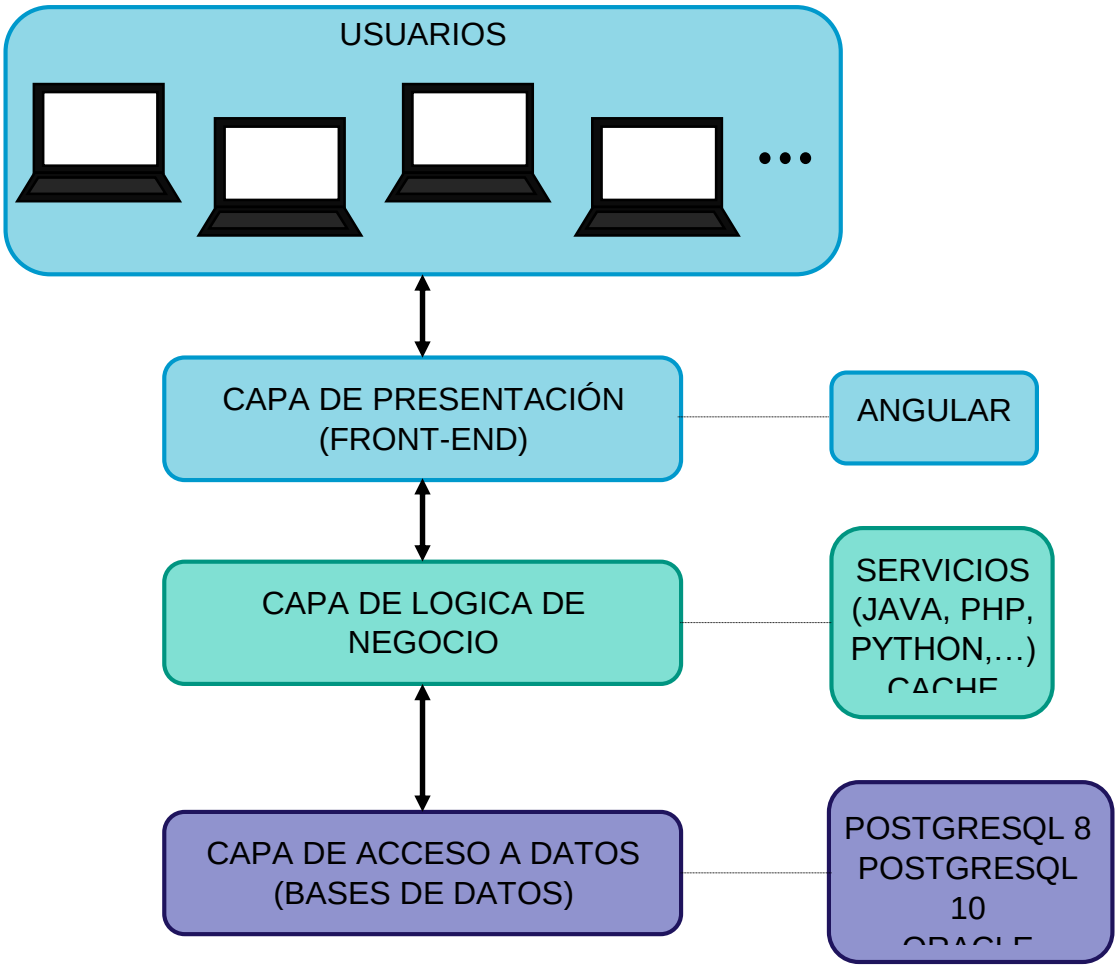
4.1 ARQUITECTURA DEL SISTEMA

La arquitectura del sistema hace referencia a la manera en la que este se estructura, generando un impacto ya sea positivo o negativo en la calidad del software, teniendo en cuenta que los posibles criterios de calidad pueden ser mantenibilidad, usabilidad, desempeño, entre otros. Un mal diseño del sistema implicaría demoras en los tiempos de respuesta, dificultando que el usuario final manipule el sistema, así como alto acoplamiento que conlleva en el peor de los casos a que se sea imposible la modificación de funcionalidades existentes o la creación de nuevas funcionalidades. En otras palabras, la arquitectura del sistema trata de realizar una organización global de este. Existen diversas arquitecturas de software como lo son las monolíticas, cliente-servidor, en capas, entre otras.

El sistema diseñado adopta una arquitectura en tres capas, en la que los datos, la lógica de la aplicación y la interfaz de usuario están separados, las capas hacen referencia a la separación lógica de la aplicación. La primera es la capa de presentación, encargada de la interacción entre el usuario y el sistema, esta corresponde al f*ront-end* que es responsable de todo lo relacionado con la interfaz con la que el usuario final interactúa, generalmente se manejan tres tecnologías en la construcción de este, las cuales son HTML, CSS y JavaScript, y a su vez en el mercado actual existen tecnologías que las apoyan como es el caso de los *frameworks* y librerías, un gran ejemplo de ello, es *frameworks,* como Angular y librerías tales como *Bootstrap* o *Animated*. La segunda capa es la encargada de la lógica del negocio, es decir, es la que establece las reglas de funcionamiento del sistema, esta capa corresponde al back-end, los lenguajes de programación

utilizados para el desarrollo de este pueden ser: PHP, Java, Phyton, ASP.NET, Node.js, entre otros. En esta capa se hace uso de los servicios web basados en REST, realizando la comunicación de este con el front-end mediante el protocolo HTTP que se encarga del intercambio de los datos, la identificación de los recursos se realiza a través del uso de URLs que corresponden a la localización de dicho recurso, la forma de representar estos es por medio del formato JSON (*JavaScript Object Notation*) puesto que proporciona alta velocidad de procesamiento, menor tamaño de los archivos, y es un formato bastante simple; además la lógica de negocio también tiene en cuenta códigos de estado, métodos y cabeceras HTTP. La tercera y última capa es la de acceso a los datos, que almacena toda la información correspondiente a la aplicación, esta se encarga de la persistencia de los datos. En la figura 59 se evidencia gráficamente la arquitectura del sistema diseñado

Figura 59. Arquitectura del sistema



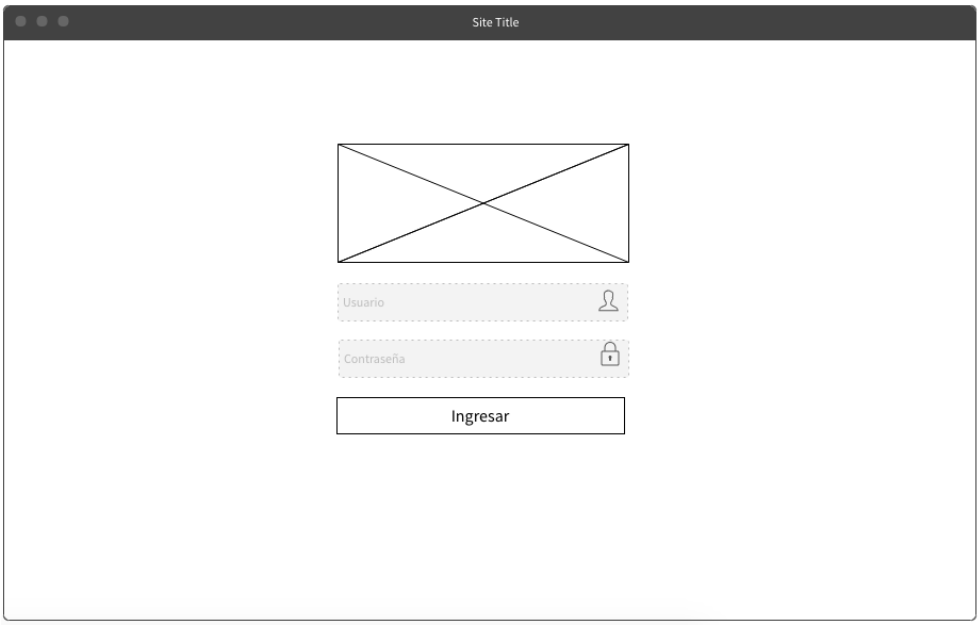
Usar esta arquitectura permite la clara separación entre el front-end y el back-end de un sitio web, que trae consigo gran cantidad de ventajas, entre ellas se tiene que el mismo back-end puede ser utilizado para el desarrollo del front-end sobre diferentes marcos de trabajo, otra ventaja se relaciona con la reducción de tiempo de desarrollo ya que estos dos pueden ser desarrollados por equipos de trabajo completamente diferentes. Otro punto a tener en cuenta dentro de cualquier arquitectura es el tema de los niveles, que son la separación física del sistema. En este caso se tienen múltiples niveles puesto que se encuentra diseñada de manera que pueda crecer en una diversa cantidad de servidores o equipos. Inicialmente se está trabajando con tres niveles que corresponden a cada una de las capas, pero se proyecta que con el tiempo estos aumenten, dependiendo de la disponibilidad de

recursos y de las necesidades. Por último cabe destacar que la arquitectura de esta aplicación tiene un enfoque orientado a servicios, puesto que está, en la capa de lógica de negocio se compone de servicios, los cuales pueden estar construidos bajo diferentes lenguajes de programación además de ser independientes unos de otros, heredando así las ventajas que esta arquitectura trae consigo, como por ejemplo mayor escalabilidad, interoperabilidad, mantenibilidad, entre otros.

4.2 DISEÑO DE LA INTERFAZ DE USUARIO

A continuación, se presenta la maquetación de las pantallas principales del sistema mediante *wireframe*, sirviendo estas como base para el desarrollo del prototipo. La figura 60 muestra la maqueta de la pantalla de inicio de sesión del sistema. En esta se debe ingresar el usuario y la contraseña en los campos correspondientes.

Figura 60. Maqueta de la pantalla de inicio de sesión

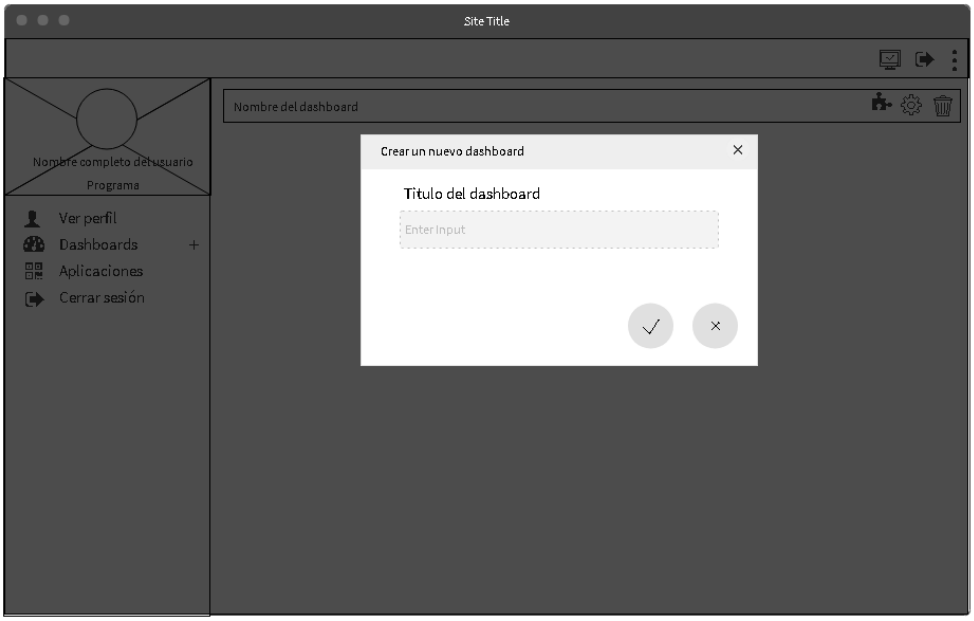


En la figura 61 se presenta el boceto de la pantalla, que el usuario visualiza cuando ingresa por primera vez al sistema. En esta se tiene un *dashboard* creado por defecto, con las opciones de crear, configurar y eliminar *dashboard*, además de agregar widgets.

Figura 61. Maqueta de la pantalla principal del sistema

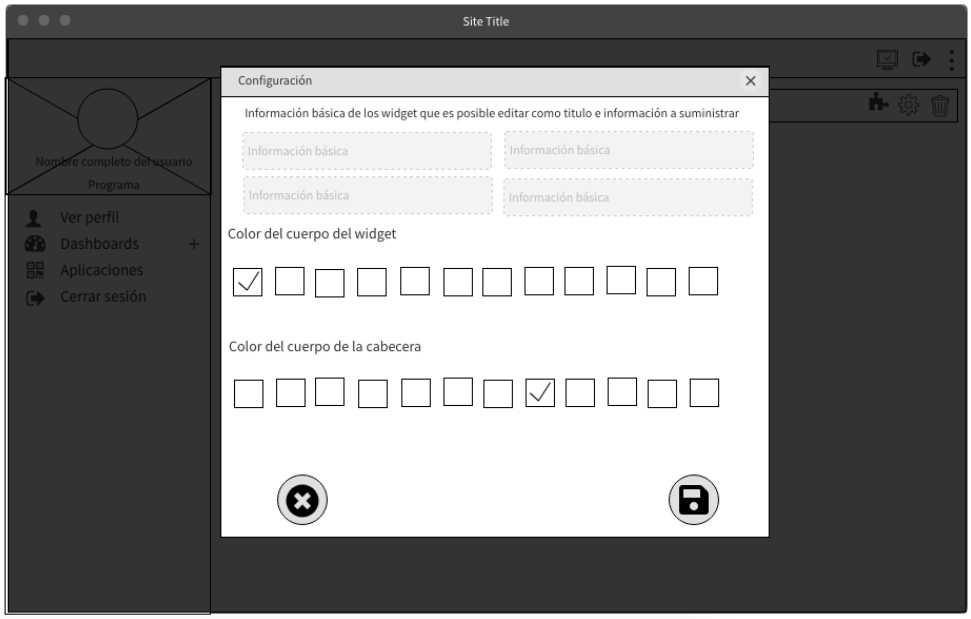
La figura 62 muestra la pantalla que permite la creación de un nuevo *dashboard*. En el formulario se debe ingresar el título que este tendrá.

Figura 62. Maqueta del formulario de creación de dashboard del sistema



Otro boceto de formulario es el relacionado con las configuraciones de los widgets. En la figura 63 se presenta la maqueta para este

Figura 63. Maqueta del formulario de configuraciones del widget

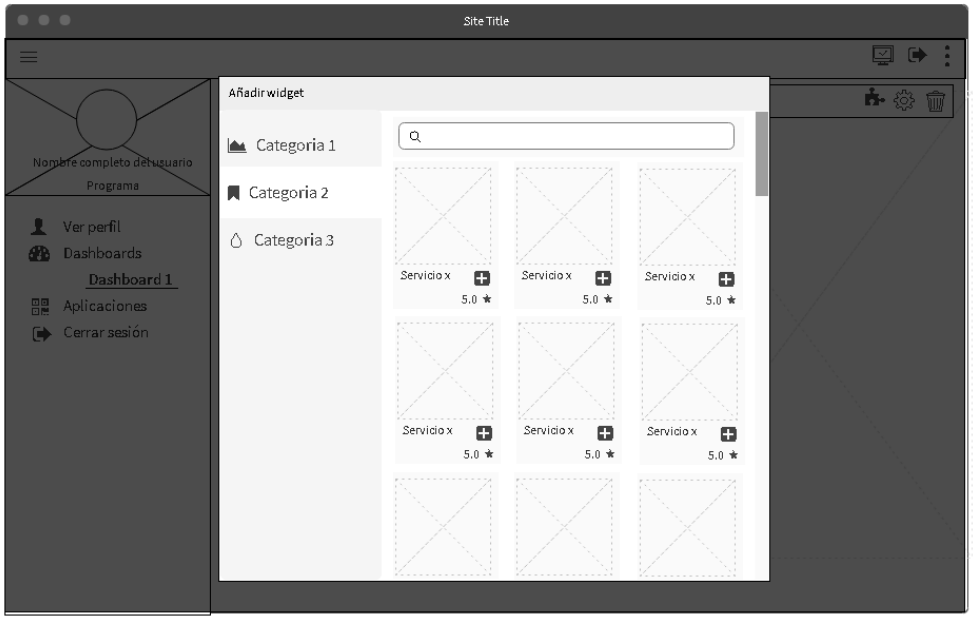


El sistema cuenta con alertas que avisan al usuario cuando va a realizar una acción, como eliminar un dashboard o alguno de los widgets que tenga agregados. En la figura 64 se muestra el boceto de estas alertas, tomando como ejemplo la eliminación de un dashboard

Figura 64. Maqueta de pantalla de alerta del sistema

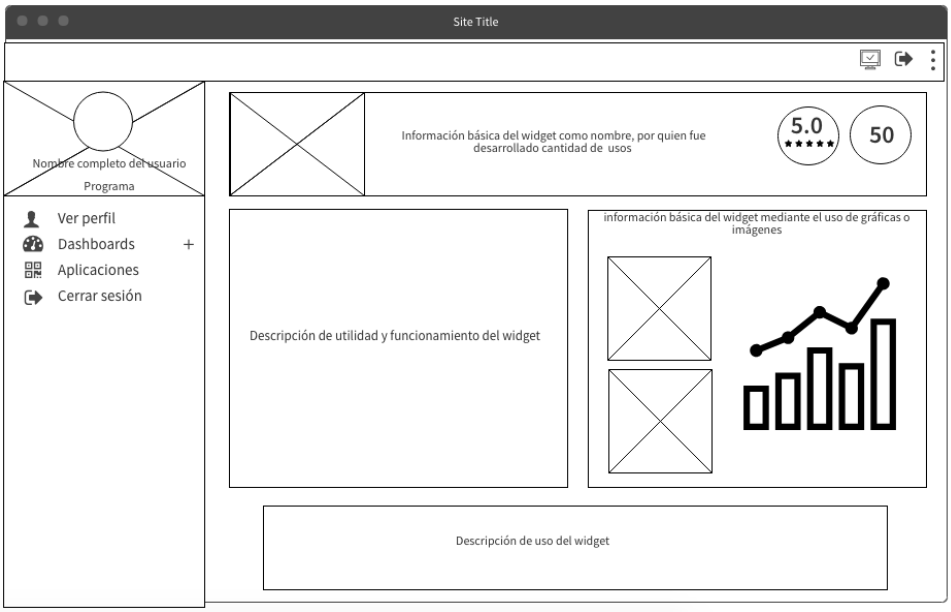
En la figura 65 se presenta la forma de visualización de los widgets disponibles en el sistema. Cada uno de ellos cuenta con la imagen, el nombre y la calificación respectiva. Además, en esta pantalla se tiene la posibilidad de agregar y de ver información detallada de estos.

Figura 65. Maqueta de pantalla de visualización de widgets disponibles



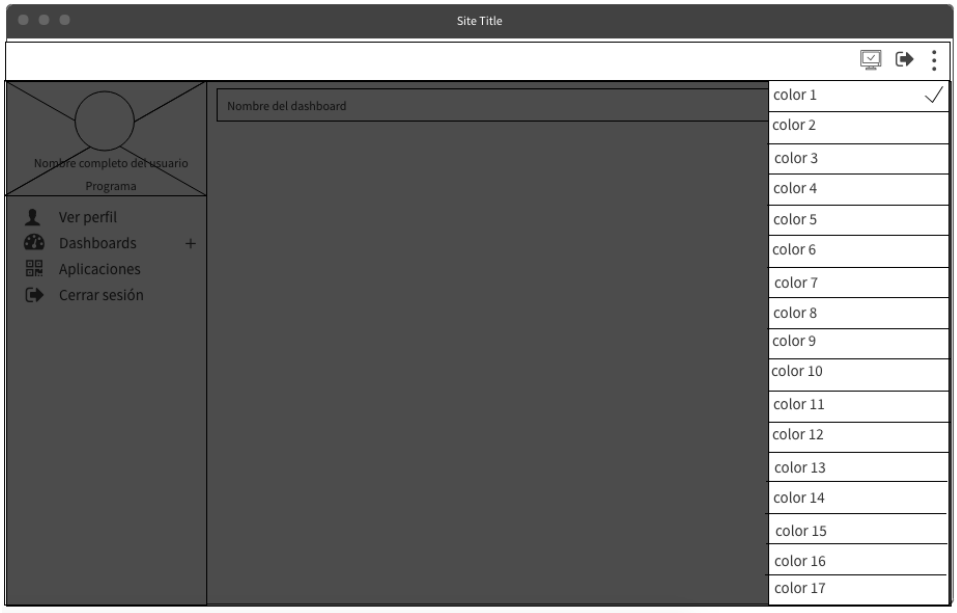
El sistema permite visualizar información correspondiente a cada uno de los widgets. Por ejemplo, miniaturas, descripción, autor, estadísticas de uso y calificaciones del widget. El boceto de esta pantalla se puede observar en la figura 66.

Figura 66. Maqueta de la pantalla de información del widget



La figura 67 muestra la maqueta de la pantalla de cambiar tema de la interfaz. En esta se permite elegir uno de una serie de temas disponibles.

Figura 67. Maqueta de la pantalla cambiar tema de la interfaz



5 PROTOTIPO

5.1 TECNOLOGIAS UTILIZADAS

5.1.1 Angular

Este *framework* se utilizó para el desarrollo del *front-end* del prototipo, puesto que es un marco de trabajo que permite crear aplicaciones en el lado del cliente, con el fin de aprovechar sus recursos y no recargar el servidor. Fue desarrollado por Google, y es mantenido por esta misma compañía, la cual se encarga de introducir mejoras y de respaldarlo. Está escrito en *TypeScript,* cuyo lenguaje base es JavaScript y es mantenido por Microsoft, cuenta con un tipado estático y herramientas de programación orientada a objetos además tiene una característica muy importante y es que convierte su código en código JavaScript el cual es el que interpreta el navegador.

El patrón de diseño que utiliza Angular es MVVM (*model–view–viewmodel* o modelo-vista-modelo de vista), su objetivo es separar la interfaz de usuario de la lógica de negocio. El modelo se refiere al acceso a datos, la vista se encarga de todo lo relacionado con la interfaz de usuario y el modelo de vista tiene que ver con el enlace entre la vista y el modelo. Los actores principales de una aplicación desarrollada con Angular son los componentes, los módulos y los servicios. Los componentes son pequeñas clases que son identificadas por medio del decorador @Component(), los cuales se encargan de cumplir una tarea específica, estos están relacionados directamente con su template o plantilla HTML, en una aplicación se

tienen múltiples componentes, por ejemplo el menú lateral sería un componente, otro seria la cabecera de la aplicación, entre otros. En la figura 68 se muestra un ejemplo de cómo sería un componente en Angular

Figura 68. Ejemplo de un componente en Angular

```
1. import { Component } from '@angular/core';
2.
3. @Component({
4.   selector: 'app-root',
5.   template: `
6.     <h1>{{title}}</h1>
7.     <h2>My favorite hero is: {{myHero}}</h2>
8.     `
9. })
10. export class AppComponent {
11.   title = 'Tour of Heroes';
12.   myHero = 'Windstorm';
13. }
```

Fuente 13. ANGULAR. Src/app/app.component.ts [imagen]. Displaying Data. [Consultado: 10 de noviembre de 2018]. Disponible en internet: https://angular.io/guide/displaying-data

Los módulos se encargan de declarar y agrupar un conjunto de componentes, cada aplicación tiene múltiples módulos, teniendo uno principal o raíz que por convención se denomina AppModule, este actor hace uso del decorador @NgModule(). A continuación, en la figura 69, se presenta la estructura básica de un módulo en Angular

Figura 69. Ejemplo de un módulo en Angular

```
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
@NgModule({
  imports:      [ BrowserModule ],
  providers:    [ Logger ],
  declarations: [ AppComponent ],
  exports:      [ AppComponent ],
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

Fuente 14. ANGULAR. Src/app/app.module.ts [imagen]. Introduction to modules. [Consultado: 10 de noviembre de 2018]. Disponible en internet: https://angular.io/guide/architecture-modules

Los servicios son clases que se encargan de la lógica de la aplicación, utilizan el decorador @Injectable() y su tarea principal es obtener los datos del servidor. La estructura básica de estos se presenta en la figura 70

Figura 70. Ejemplo de un servicio en Angular

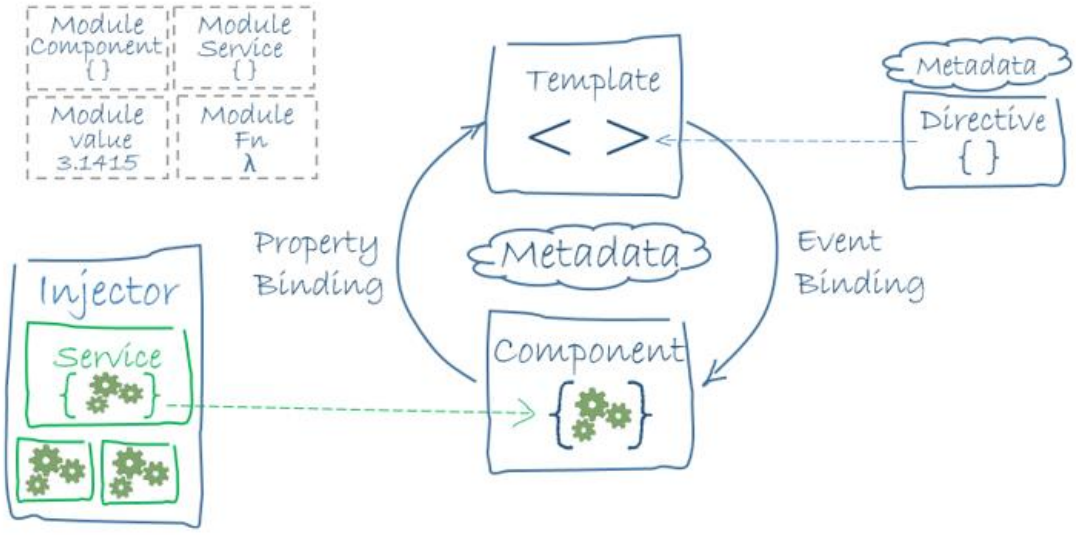
```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root',
})
export class HeroService {
  constructor() { }
}
```

Fuente 15. ANGULAR. Src/app/heroes/hero.service.ts [imagen]. Dependency Injection in Angular. [Consultado: 10 de noviembre de 2018]. Disponible en internet: https://angular.io/guide/dependency-injection

En la figura 71 se resume el funcionamiento de este framework

Figura 71. Arquitectura de Angular



Fuente 16. ANGULAR. Architecture overview [imagen]. [Consultado: 10 de noviembre de 2018]. Disponible en internet: https://angular.io/guide/architecture

Si se quiere profundizar acerca de este tema, en la página oficial de Angular (https://angular.io/) se encuentra la documentación concerniente al *framework*.

5.1.2 Servicios web

La construcción de los servicios web, que conforman el back-end, está directamente relacionada con los casos de uso descritos en el capítulo 3 de análisis. Estos se desarrollaron mediante la última versión de java, la cual es java 8, que trae consigo mejoras respecto de las versiones anteriores, en cuanto a seguridad, eficiencia en el desarrollo y ejecución, entre otras. Estos servicios se alojaron en un servidor web Apache Tomcat 9.

5.1.3 JSON Web Token (JWT)

Es un estándar abierto que permite generar tokens, que son usados para controlar el acceso a las aplicaciones, este token se considera confiable puesto que es

firmado digitalmente. En la página oficial de JWT[51] dicen que los dos escenarios en donde estos tokens son más útiles son la autorización y el intercambio de información. En el primer escenario al inicio de sesión del usuario se recibe por parte del servidor el token que tendrá que ser enviado en las peticiones siguientes esto con el fin de garantizar la identidad del usuario, este escenario es el usado en este proyecto, el prototipo realizado funciona mediante este enviando el token cada vez que se hacen peticiones a cada servicio. El segundo escenario es usado cuando se necesita compartir información de forma segura. Toda la información concerniente a este estándar se puede encontrar en el RFC 7519 o en la página oficial https://jwt.io/.

5.1.4 Bases de datos

PostgreSQL es uno de los gestores de bases de datos más utilizados, es gratuito y de código abierto. Algunas de sus características son la estabilidad y confiabilidad que este provee, así como reducción de costos y gran variedad de tipos nativos. PostgreSQL 10 fue lanzado el 5 de octubre de 2017 trayendo consigo gran cantidad de características, haciendo que se considere una de las actualizaciones más amplias. Entre estas características se puede destacar que ahora se pueden realizar consultas en paralelo de forma más eficaz, se mejoró la seguridad y corrección de errores. Para el desarrollo de este prototipo se usaron dos versiones de este sistema de gestión de base de datos, en primer lugar se usó PostgreSQL 8 que contiene el modelo de datos de la Universidad de Pamplona; y en segundo lugar se utilizó PostgreSQL 10 para todo lo concerniente al modelo de datos de esta nueva aplicación.

5.1.5 Patrón cache

El término cache se puede definir como un almacén de datos temporal en el cual se guardan copias de otros datos que serían los originales, en esta, el acceso es mucho más rápido. Una de las mayores ventajas de esta es que aporta escalabilidad

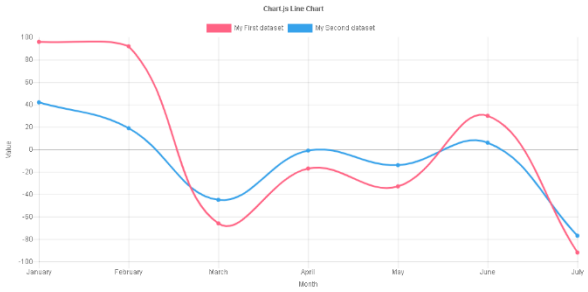
[51] AUTH0. Introduction to JSON Web Tokens. JWT [en línea]. [Consultado: 05 de octubre de 2018]. Disponible en internet: https://jwt.io/introduction/

a los sistemas, además de disminuir la latencia, el consumo de ancho de banda y de mejorar los tiempos de carga.

5.1.6 Librerías

En el desarrollo del prototipo se utilizaron cinco librerías: Chart.js, Sweet alert.js, Compodoc, Animate.css y Angular2gridster. Chart.js es una librería de JavaScript, fácilmente integrable con Angular, que permite realizar gráficos responsivos y animados, cuenta con diferentes opciones de configuración que proporcionan diversos tipos de gráficas, los cuales se pueden usar dependiendo de las necesidades que se tengan. En su página web, https://www.chartjs.org/, se encuentra toda la información necesaria para utilizarla. En la figura 72, se presenta un ejemplo de grafica de líneas, realizado con esta librería

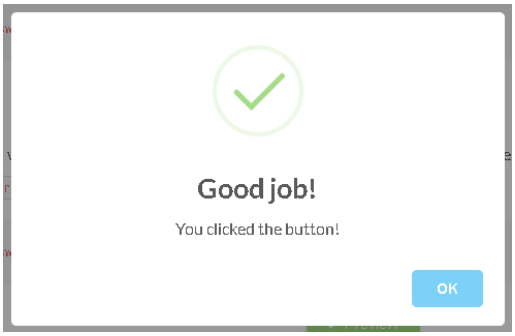
Figura 72. Ejemplo de una gráfica de líneas con la librería Chart.js



Fuente 17. CHART.JS. Chart.js Line Chart [imagen]. Samples. [Consultado: 10 de noviembre de 2018]. Disponible en internet: https://www.chartjs.org/samples/latest/charts/line/basic.html

Al igual que Chart.js, Sweet alert.js, es una librería desarrollada en JavaScript que brinda la posibilidad de mostrar mensajes emergentes de una forma fácil y bonita, haciendo que esta se haya popularizado en la actualidad. Un ejemplo de estas alertas se evidencia en la figura 73

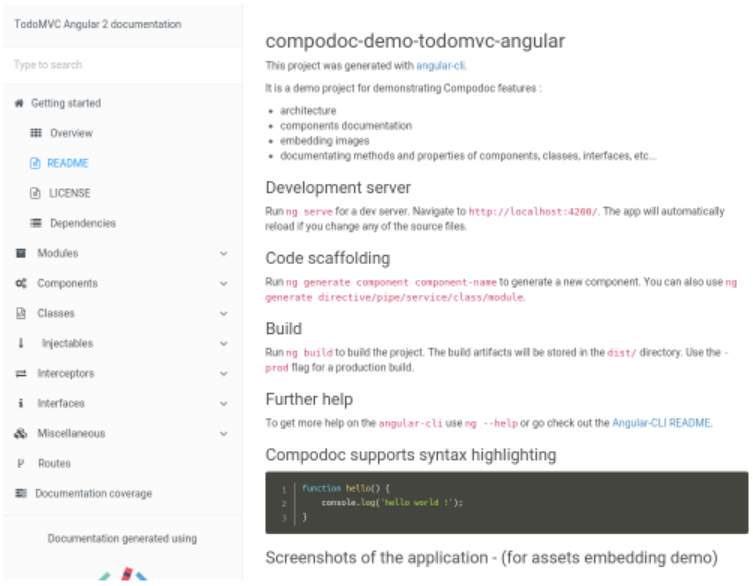
Figura 73. Ejemplo alerta con Sweet alert.js.



Fuente 18. EDWARDS, Tristan. [Imagen]. Installation. [Consultado: 10 de noviembre de 2018]. Disponible en internet: https://sweetalert.js.org/guides/#installation

Compodoc es una herramienta que permite generar la documentación de un proyecto en Angular, además de tener un diseño limpio y simple, es de código abierto, su instalación es sencilla, se realiza mediante npm y es una herramienta local, es decir, que no se hace necesario el uso de un servidor para esta. En la figura 74 se puede observar el tema por defecto de esta herramienta

Figura 74. Ejemplo del tema por defecto de la herramienta



Fuente 19. OGLOBLINSKY, Vincent. Default (gitbook) [imagen]. Themes. 2016. [Consultado: 10 de noviembre de 2018]. Disponible en internet: https://compodoc.app/guides/themes.html

Animate.css como su nombre lo indica es una biblioteca de CSS que permite crear gran cantidad de animaciones para las páginas web, su uso es sencillo basta con

colocar la clase *animated* seguida del efecto deseado por ejemplo *bounceIn*, en el objeto HTML a animar. En su sitio web se encuentran todas las animaciones que esta ofrece. Por último, se tiene una implementación para Angular del Gridster de JQuery, llamado Angular2gridster, que permite que los elementos deseados dentro de una página puedan ser reubicados de acuerdo a los deseos del usuario.

5.2 RESULTADOS

El termino validación hace referencia al modo de constatar que todo lo especificado cumple con lo que el cliente deseaba o no. La idea es realizar una evaluación del sistema durante su desarrollo o cuando este ya haya culminado, esto con el fin de definir si satisface o no los requisitos. Esta validación se puede abordar desde diferentes criterios, los cuales pueden ser usabilidad, funcionalidad y escalabilidad, entre otros. El prototipo realizado en este proyecto se puede validar parcialmente puesto que no es un producto final y teniendo en cuenta los requerimientos que el cliente, que en este caso es el Centro de Investigación Aplicado y Desarrollo en Tecnologías de la Información (CIADTI), especificó.

La funcionalidad determina si el sistema cumple con las características para las que ha sido diseñado. En este punto el enfoque que se le da es relacionado con el cumplimiento de algunos de los requerimientos, cuando se habla de algunos y no de todos es porque se está tomando como referencia un prototipo más no un producto final. A continuación, se presenta la descripción funcional del prototipo.

El prototipo resultante de este proceso es una aplicación web en la que el usuario debe iniciar sesión para acceder a las funcionalidades que esta le brinda. La figura 75 muestra la página de inicio de sesión

Figura 75. Página de inicio de sesión

Una vez el usuario ha sido validado, el sistema permite que este agregue la cantidad de *dashboards* deseados a su interfaz, los configure o los elimine según sus necesidades, en la figura 76 se observa la página principal del sistema, con cada una de las opciones mencionadas anteriormente. Al agregar un nuevo *dashboard* se pide el título que este tendrá, como se observa en la figura 77. Si se desea modificar después de agregado, lo que se actualiza será este título. En cada uno de los *dashboards* se puede agregar la cantidad deseada de componentes denominados widgets. Al oprimir la opción agregar widgets se muestra un listado de los que se encuentran disponibles, esto se representa en la figura 78. En este listado se tiene la opción de visualizar la información concerniente al widget, que incluye la descripción, las calificaciones dadas, entre otros; o de agregarlo. La figura 79 presenta la pantalla de información del widget

Figura 76. Pantalla de inicio de la aplicación

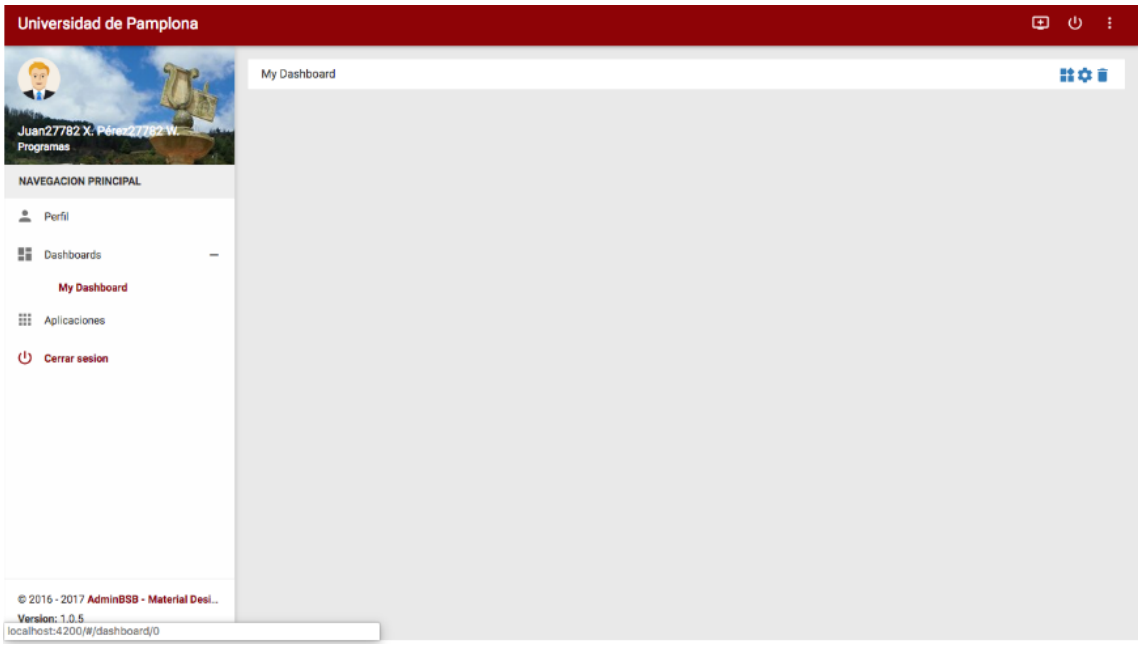


Figura 77. Pantalla de creación de un nuevo dashboard

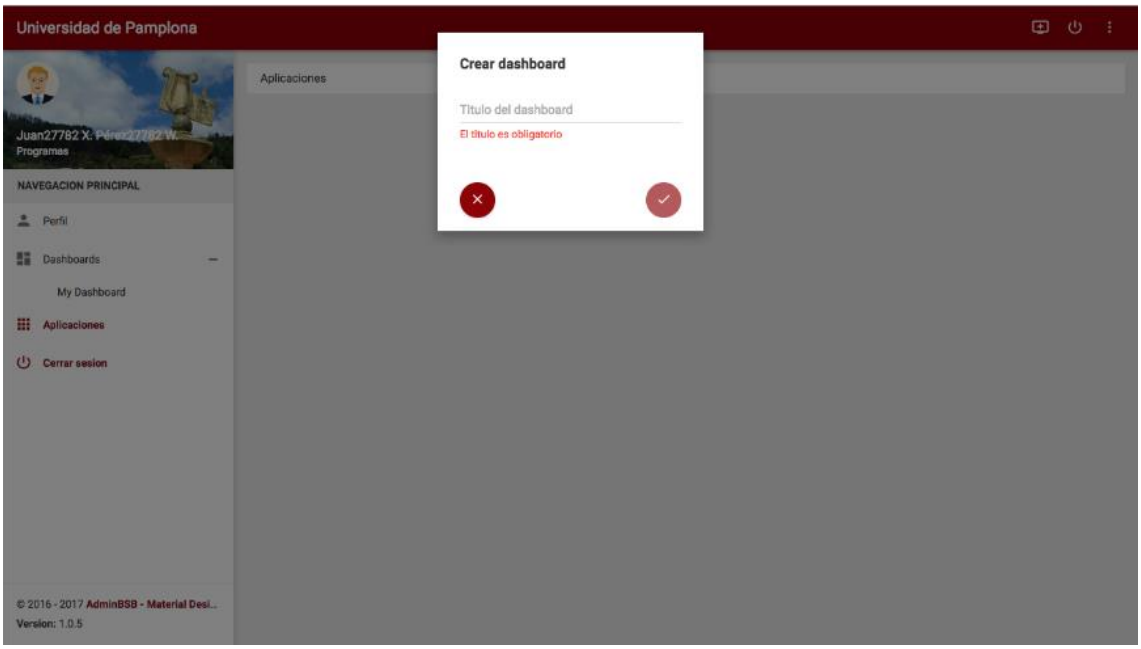


Figura 78. Listado de widgets disponibles

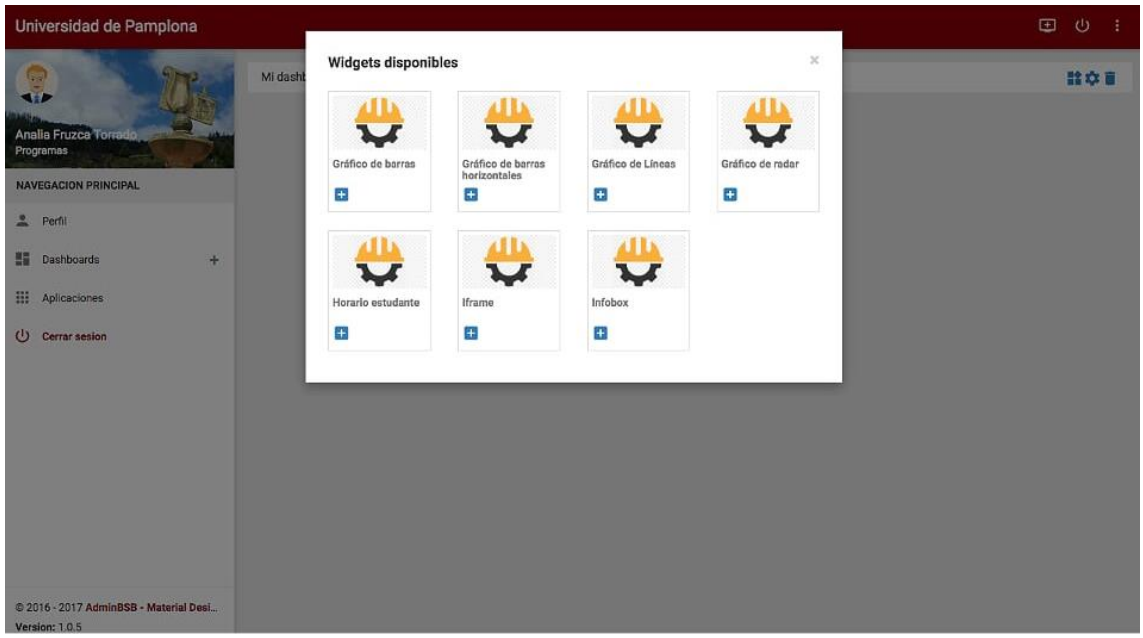


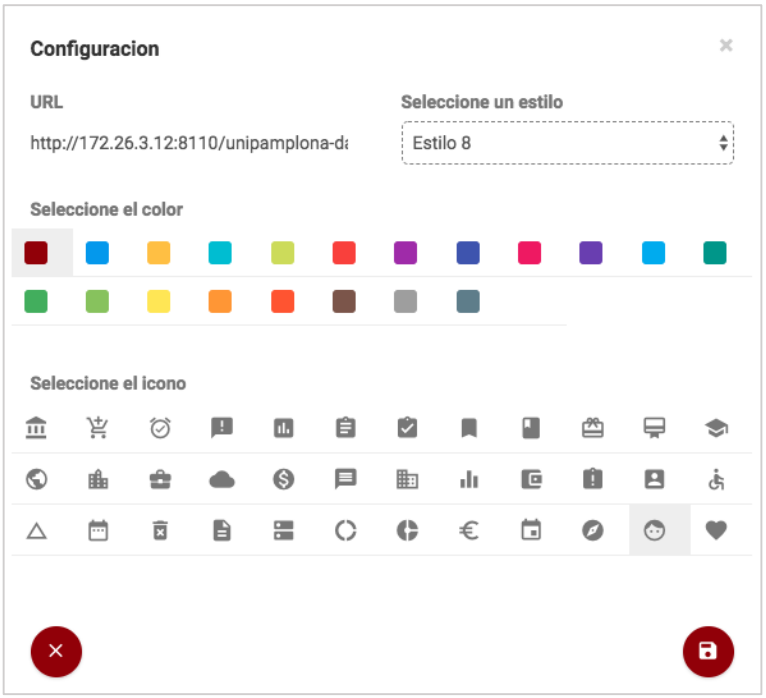
Figura 79. Ventana de información del widget

Los widgets se dividen en cuatro tipos:

Infobox: Este tipo de widget se encarga de representar gráficamente un título y un valor, estos datos se consumen de un servicio que retorna esta información en una estructura tipo JSON, un ejemplo de este tipo se puede observar en la figura 80. Este componente se configura ingresando la URL del servicio, seleccionando su forma de visualización de entre un conjunto de 15 estilos, su color y su icono. La figura 81 muestra el formulario de configuración de este widget

Figura 80. Ejemplo del widget infobox

Figura 81. Formulario de configuración del widget infobox



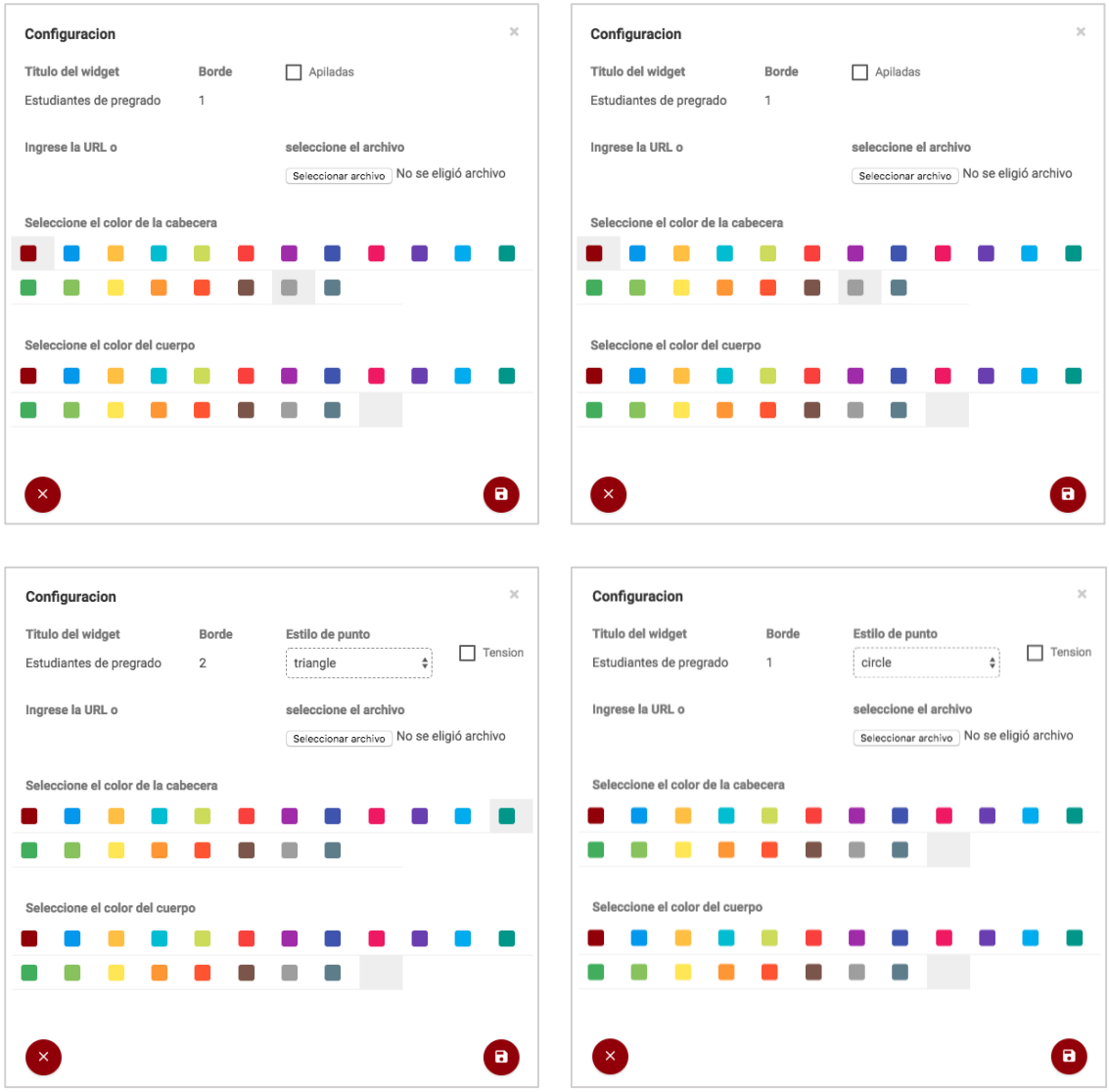
Graphic: Permite realizar graficas estadísticas mediante la lectura de una URL que corresponde a un servicio que retorne la información requerida para realizar dicha grafica o por medio de un archivo CSV el cual debe tener una estructura definida.

En este tipo de widget se identificaron cuatro subtipos los cuales corresponden a las diferentes graficas que se pueden realizar, estos son: grafica de barras, grafica de barras horizontales, grafica de líneas y grafica de radar. En la figura 82 se observa un ejemplo de cada una de las gráficas que se pueden realizar con este widget.

Figura 82. Ejemplo de los subtipos de graficas de barras, barras horizontales, lineas y radar respectivamente

Al realizar la configuración de estas, además de ingresar ya sea una URL o el archivo CSV, se permite ingresar un título que es mostrado en la cabecera del widget, seleccionar un color para el encabezado y otro para el cuerpo, así como otras preferencias como tensión, ancho del borde, estilo del punto, entre otras. A continuación, en la figura 83 se presentan los formularios de configuración de las graficas

Figura 83. Formularios de configuración de los subtipos de graficas de barras, barras horizontales, lineas y radar respectivamente



Al visualizar la información de este widget además se puede encontrar el formato del archivo CSV que se debe subir para poder realizar dichas gráficas. La figura 84 muestra un ejemplo del formato de este archivo

Figura 84. Formato del archivo CSV para realizar las graficas

Iframe: Este widget permite visualizar un sitio web, en la figura 85, se muestra un ejemplo de ello, mediante la página de la Universidad de Pamplona y de Angular. Al configurar este tipo, se requiere que sea ingresada una URL que será la dirección web mostrada en el widget, un título y seleccionar un color para el encabezado y otro para el cuerpo. La figura 86 representa el formulario de configuración del widget iframe

Figura 85. Ejemplos del widget iframe

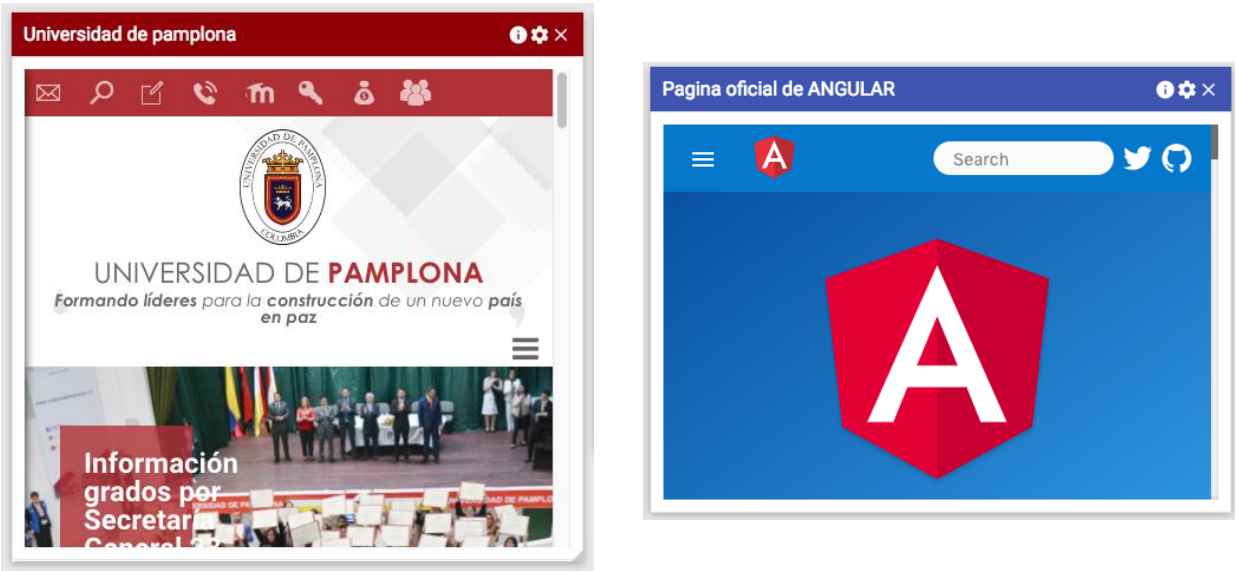
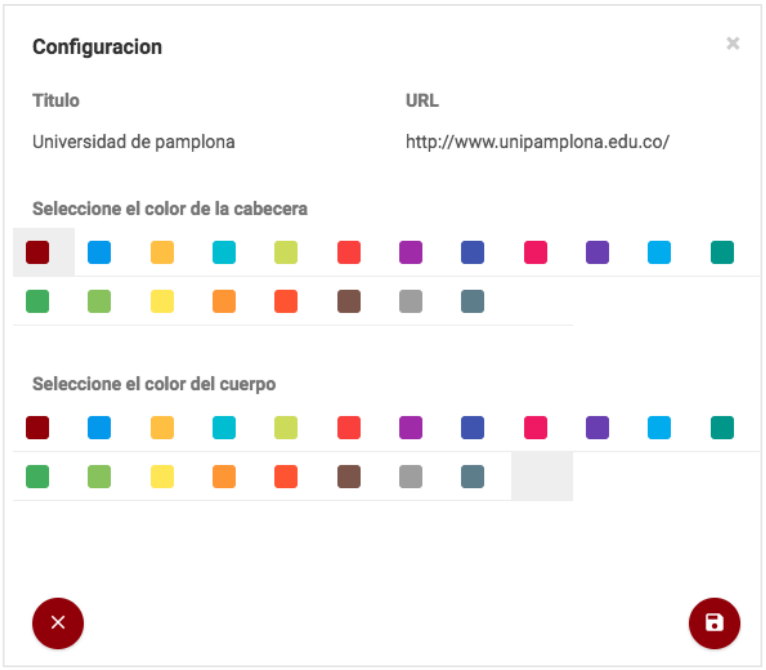


Figura 86. Formulario de configuración del widget iframe



Application: Este último tipo se encarga de visualizar los servicios que son disponibles para una mayor cantidad de personas, por ejemplo, el horario del estudiante, como se muestra en la figura 87, o del docente, las calificaciones actuales, deudas en la biblioteca, entre otros. Al agregar este widget se configura un título y se selecciona el color del encabezado y del cuerpo del mismo, este formulario se puede observar en la figura 88.

Figura 87. Ejemplo de widget application

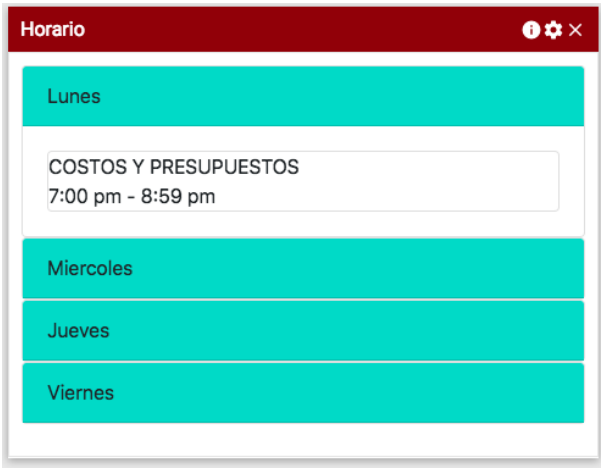
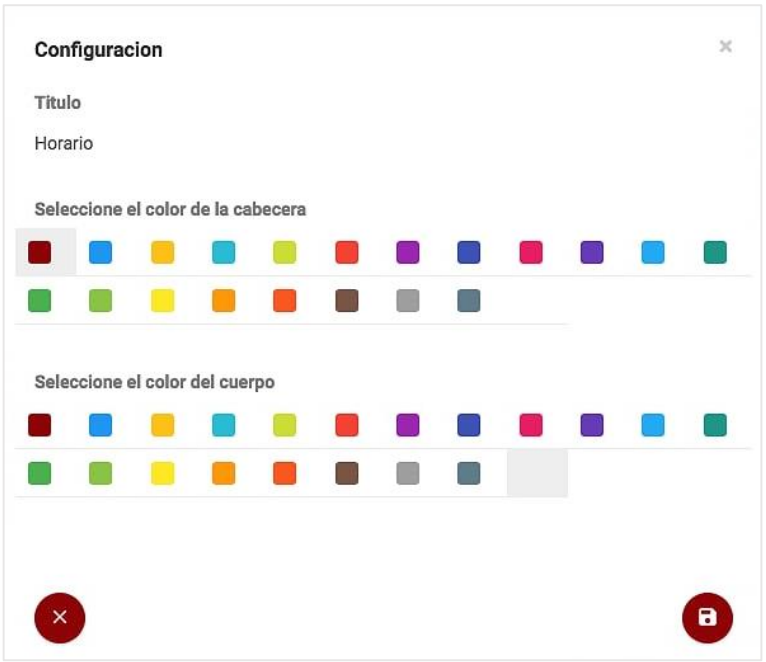


Figura 88. Formulario de configuración del widget application



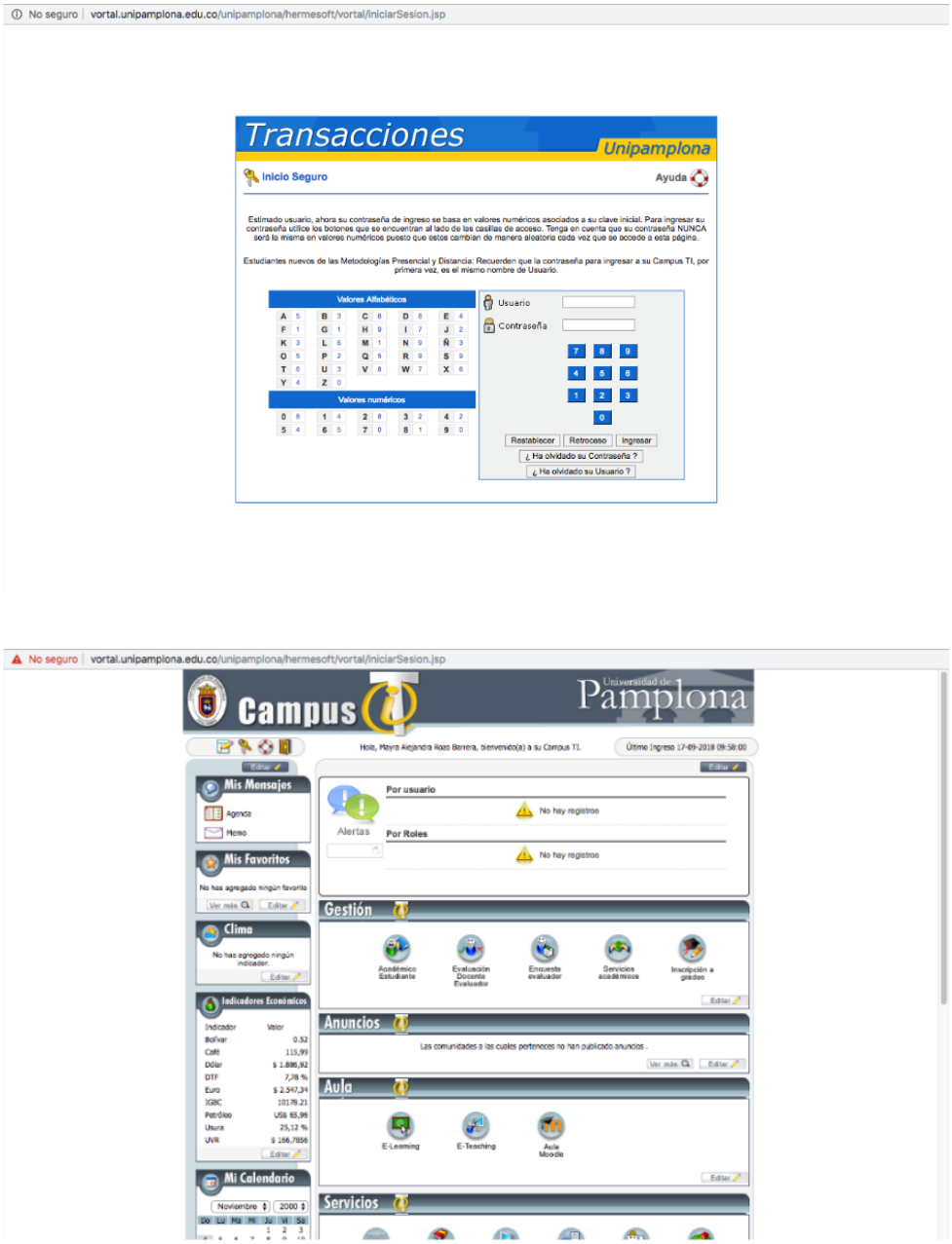
Como se mencionó anteriormente los widgets pueden ser agregados, modificados, así como eliminados del dashboard deseado. Otra característica que brinda el prototipo es que el usuario puede cambiar el tema de colores de la aplicación. La figura 89 presenta la pantalla de cambio de tema de la interfaz

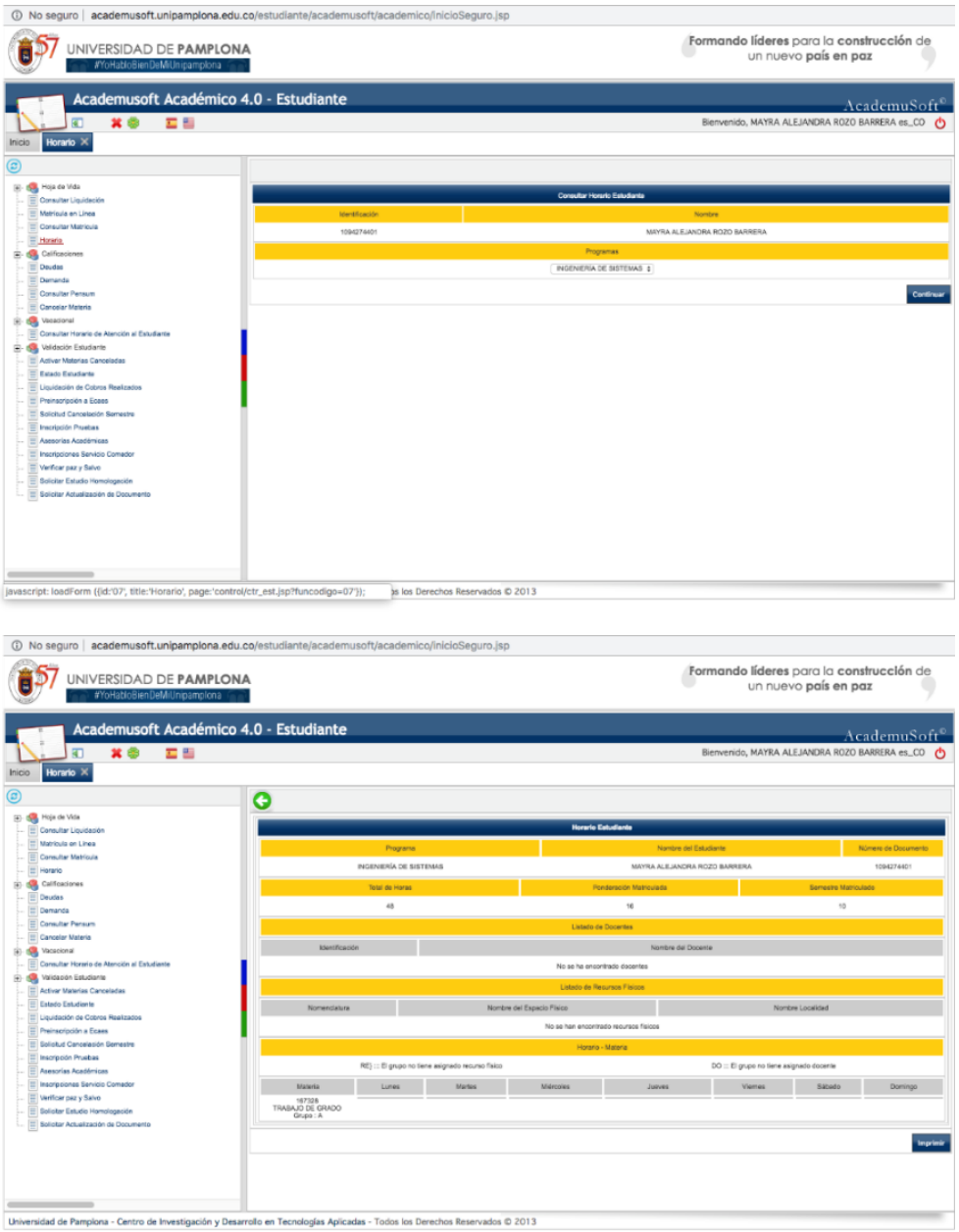
Figura 89. Menú de cambio de tema

Conociendo estas características del prototipo realizado, cabe destacar que esta interfaz permite que cada uno de los usuarios la personalice a su gusto, conllevando así a que cada actor del sistema tenga su interfaz completamente diferente a la de los demás. Como se observa en esta descripción, los requerimientos 1, 2, 3, 4, 5, 6, 7, 8, 10 y 19 que se encuentran descritos en la sección de análisis, se están cumpliendo. Otro criterio es la usabilidad que está directamente relacionada con la experiencia de usuario, existen diversos factores que la afectan para bien o para mal, entre ellos se tiene la cantidad de clics que debe dar el usuario para llegar a su objetivo. El acceso al sistema puede ser tedioso si el usuario debe realizar demasiadas acciones para acceder a las funcionalidades, teniendo en cuenta que de esto depende la satisfacción de este, por ejemplo, si un estudiante desea consultar su horario, en el sistema actual el proceso que debe seguir es que debe iniciar sesión, luego oprimir el botón académico estudiante, para así poder buscar entre el menú lateral izquierdo la opción que en este caso es horario, dar clic en este para que le aparezca una ventana en donde deberá seleccionar el programa y

por ultimo dar clic en continuar, de esta manera el sistema visualiza el horario. En la figura 90 se muestra este proceso:

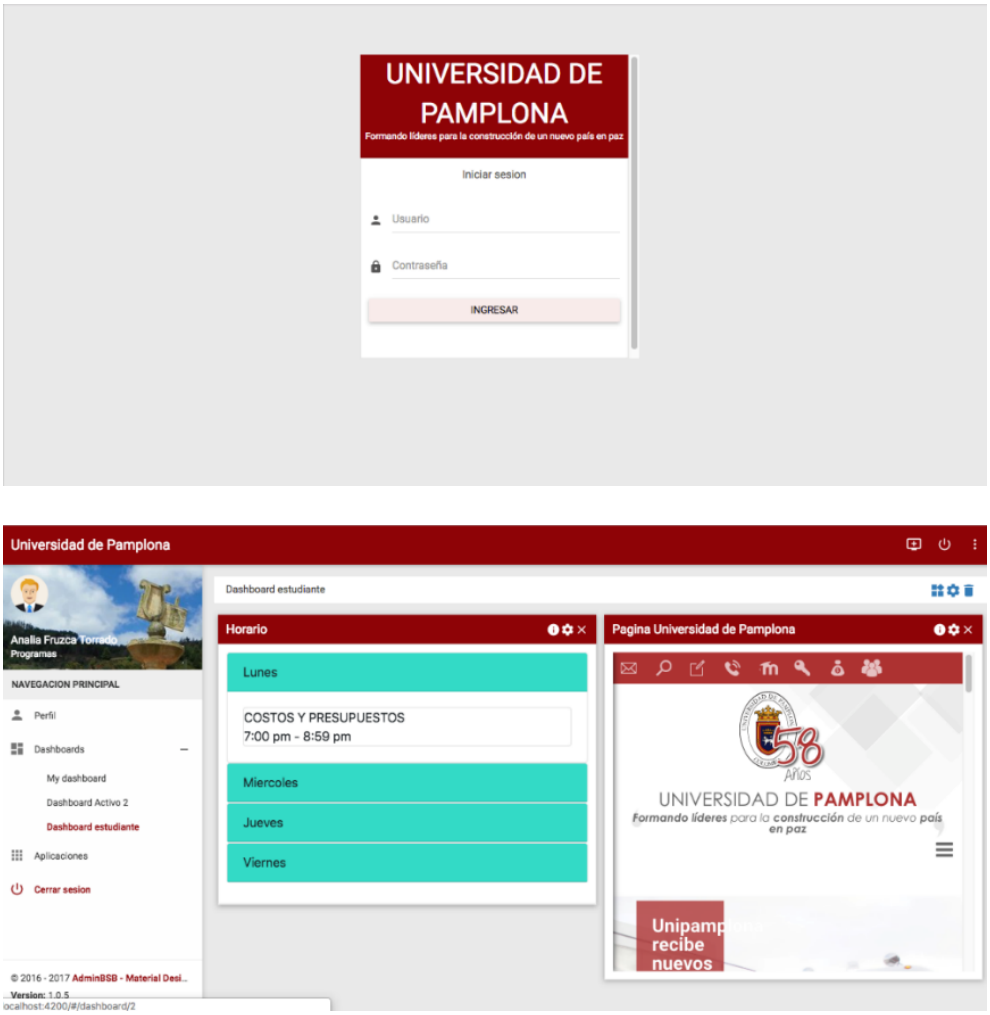
Figura 90. Pasos para visualizar el horario de un estudiante en el sistema actual





Ya se explicó los pasos a seguir en el sistema actual, ahora se aborda este mismo problema en el prototipo realizado. En este caso el usuario debe iniciar sesión, buscar en el menú lateral el *dashboard* en el que tiene agregado el horario, dar clic en este y de esta forma aparecerá lo que el estudiante quería visualizar. En la figura 91 se muestra este proceso

Figura 91. Pasos para visualizar el horario de un estudiante en



Comparando este proceso en los dos sistemas, se tiene que la estrategia propuesta reduce significativamente esta cantidad de acciones que el usuario debe realizar para lograr lo deseado.

6 CONCLUSIONES

- La investigación realizada logro proponer el diseño de una estrategia que permite realizar la visualización de servicios, mediante un *dashboard,* cumpliendo con características de usabilidad, escalabilidad e interoperabilidad.
- En el marco de referencia se profundizo acerca de las arquitecturas orientadas a servicios y servicios web basados en diferentes estilos. De esta manera se observó que el estilo REST es uno de los más usados en la actualidad gracias a sus innumerables beneficios.
- El uso de servicios web basados en REST trae múltiples ventajas como facilidad de uso, comunicación por medio del protocolo HTTP, así como bajo acoplamiento entre los componentes de un sistema. Cabe destacar que también provee interoperabilidad, permitiendo que su construcción se haga mediante el uso de diferentes lenguajes de programación.
- Los requerimientos funcionales del sistema se obtuvieron mediante mesas de trabajo con los ingenieros del CIADTI, de los cuales se lograron representar diez en el prototipo realizado.
- Se definió para el sistema una arquitectura de tres capas, ya que provee un aumento de la escalabilidad, fácil mantenimiento del software y separación clara del front-end y el back-end.
- Se estableció inicialmente una arquitectura de tres niveles (forma en que las capas lógicas se encuentran distribuidas en forma física), los cuales soportan cada una de las capas del sistema, con la proyección que estos puedan aumentar. Gracias a la arquitectura planteada el sistema puede creer con base a las necesidades y a los recursos con los que cuente la institución.
- El prototipo desarrollado, permite a los usuarios a personalizar su entorno, buscando de esta manera que la interfaz se adecúe a sus gustos y necesidades, Ofreciéndoles la gestión de dashboards, widgets, asignación de estilos y demás características con las que cuenta el sistema.

- El uso de framework como Angular, aporta una fácil estructuración del proyecto propiciando así su clara comprensión, siendo esto ventajoso a la hora de agregar o de crear nuevas funcionalidades en un proyecto de desarrollo.

7 BIBLIOGRAFIA

BAQUERO, José. ¿Qué son los web services y qué tecnología usar en su desarrollo? ARSYS [en línea]. 3 de agosto de 2015 [consultado: 24 de julio de 2018]. Disponible en internet: https://www.arsys.es/blog/programacion/diseno-web/web-services-desarrollo/

DEPARTAMENTO DE CIENCIA DE LA COMPUTACION E INTELIGENCIA ARTIFICIAL. Introducción a los Servicios Web. Invocación de servicios web SOAP. Universidad de Alicante [en línea]. 26 de junio de 2014 [consultado: 30 de julio de 2018]. Disponible en internet: http://www.jtech.ua.es/j2ee/publico/servc-web-2012-13/sesion01-apuntes.html

DIAZ, Marinela. Arquitectura orientada a servicios. La Habana.: 14 convención científica de ingeniería y arquitectura, 2008. 2 p.

GOMEZ, Wilson. Análisis comparativo entre la arquitectura modelo vista controlador (MVC) y la arquitectura orientada a servicios (SOA) en el desarrollo del sistema de información del fondo de empleados de la universidad de pamplona (FEUP). Trabajo de grado para optar por el título de ingeniero de sistemas. Pamplona.: Universidad de Pamplona. 2012. 6 p.

IBM. Service-oriented architecture. IBM Knowledge Center [en línea], 11 de abril de 2018. [Consultado: 15 de mayo de 2018]. Disponible en internet: https://www.ibm.com/support/knowledgecenter/es/SSFTDH_7.5.1/com.ibm.wbpm.wid.main.doc/prodoverview/topics/csoa.html

IBM. Servicios web. . IBM Knowledge Center [en línea]. [Consultado: 20 de septiembre de 2018]. Disponible en internet:

https://www.ibm.com/support/knowledgecenter/es/SS7K4U_9.0.0/com.ibm.websphere.zseries.doc/ae/cwbs_wbs2.html

LATRE, Esteban. Qué es un servidor y para qué sirve [en línea]. 2016. [Consultado: 20 de septiembre de 2018]. Disponible en internet: https://infortelecom.es/blog/que-es-un-servidor-y-para-que-sirve/

LOS SANTOS, Alberto. Metodologías para el Desarrollo de Servicios en la Web: Revisión de los servicios web SOAP/REST: Características y Rendimiento [en línea]. Vigo.: Universidad de Vigo. 2009. 9 p. [consultado: 30 de julio de 2018]. Disponible en internet: http://www.albertolsa.com/wp-content/uploads/2009/07/mdsw-revision-de-los-servicios-web-soap_rest-alberto-los-santos.pdf

MARTÍNEZ, Daniel. Metodología para el diseño de Dashboards orientado hacia el registro de evidencias en el proceso de evaluaciones institucionales. Trabajo de fin de Master. Puyo.: Universidad Internacional de la Rioja. 2017. 2 p.

MERCHÁN, Ana Lucia. Importancia de utilizar Business Intelligence en una organización. Cuenca.: Universidad Tecnológica Israel. 2012. 1 p.

MORALES, Carlos. Estado del arte: Servicios Web. Bogotá.: Universidad Nacional de Colombia. 1 p.

NARVAEZ, Daniela y GONZALEZ, Sergio. Identificación de recursos web [en línea]. MDN web docs. (23 de mayo de 2018). [Consultado: 2 de agosto de 2018]. Disponible en internet: https://developer.mozilla.org/es/docs/Web/HTTP/Basics_of_HTTP/Identificaci%C3%B3n_recursos_en_la_Web

NAVARRO, Rafael. Rest vs web service. Modelado, Diseño e Implementación de Servicios Web [en línea]. 2006. 4 P. [consultado: en 30 de julio de 2018]. Disponible en internet: http://users.dsic.upv.es/~rnavarro/NewWeb/docs/RestVsWebServices.pdf

PUENTE, Raquel. Un tablero de mando para mercadeo [en línea]. En: Debates IESA. 2008, vol. XIII, no. 4, p 74-80. [Consultado: 10 de mayo de 2018]. Disponible en internet: http://servicios.iesa.edu.ve/portal/Articulos/14-Puente-Untablerodemando.pdf

ROJAS, Mauricio y MONTILVA, Jonás. Una arquitectura de software para la integración de objetos de aprendizaje basada en servicios web. Medellin.: 9th Latin American and Caribbean Conference for Engineering and Technology, 2011. 1 p.

RUELAS, Uriel. ¿Qué es RPC (llamada a procedimiento remoto)?. Coding or not [en línea]. 19 de junio de 2018. [Consultado: 29 de julio de 2018]. Disponible en internet: https://codingornot.com/que-es-rpc-llamada-a-procedimiento-remoto.

SANTAMARÍA, Carlos. Dashboard para la gestión y monitorización de cuentas del servicio cloud AWS. Grado en ingeniería telemática. Madrid.: Escuela técnica superior de ingeniería. 2017. 10 p.

VISCONTI, Marcello y ASTUDILLO, Hernán. Fundamentos de ingeniería de software [en línea]. Universidad Técnica Federico Santa María. [Consultado: 12 de octubre de 2018]. Disponible en internet: http://roa.ult.edu.cu/bitstream/123456789/401/1/08-Patrones.pdf

W3C WORKING GROUP NOTE. Web Services Architecture. W3C [en línea]. 11 de febrero de 2004 [consultado: 24 de julio de 2018]. Disponible en internet: https://www.w3.org/TR/ws-arch/#whatis

WEBDESDECERO. Wireframes: que son y cómo crearlos [en línea]. [Consultado: 20 de septiembre de 2018]. Disponible en internet: https://webdesdecero.com/wireframes-que-son-y-como-crearlos/

WIKIPEDIA. Amazon. WIKIPEDIA La enciclopedia libre [en línea]. [Consultado: 28 de julio de 2018]. Disponible en internet: https://es.wikipedia.org/wiki/Amazon

WIKIPEDIA. Coursera. WIKIPEDIA La enciclopedia libre [en línea]. [Consultado: 28 de julio de 2018]. Disponible en internet: https://es.wikipedia.org/wiki/Coursera

WIKIPEDIA. EBay. WIKIPEDIA La enciclopedia libre [en línea]. [Consultado: 28 de julio de 2018]. Disponible en internet: https://es.wikipedia.org/wiki/EBay

WIKIPEDIA. Google. WIKIPEDIA La enciclopedia libre [en línea]. [Consultado: 28 de julio de 2018]. Disponible en internet: https://es.wikipedia.org/wiki/Google

WIKIPEDIA. Grupo Santillana. WIKIPEDIA La enciclopedia libre [en línea]. [Consultado: 28 de julio de 2018]. Disponible en internet: https://es.wikipedia.org/wiki/Grupo_Santillana

WIKIPEDIA. Movistar. WIKIPEDIA La enciclopedia libre [en línea]. [Consultado: 28 de julio de 2018]. Disponible en internet: https://es.wikipedia.org/wiki/Movistar

WIKIPEDIA. Servicio web. WIKIPEDIA La enciclopedia libre [en línea]. [Consultado: 28 de julio de 2018]. Disponible en internet: https://es.wikipedia.org/wiki/Servicio_web